Constant-Size zk-SNARKs in ROM from Falsifiable Assumptions

February 5, 2024

Helger Lipmaa[1][0000−0001−8393−6821], Roberto Parisella[2][0009−0007−2241−801X], and Janno Siim[2][0000−0001−5824−7215]

[1] University of Tartu, Tartu, Estonia
[2] Simula UiB, Bergen, Norway

Abstract. We prove that the seminal KZG polynomial commitment scheme (PCS) is black-box extractable under a simple falsifiable assumption ARSDH. To create an interactive argument, we construct a compiler that combines a black-box extractable non-interactive PCS and a polynomial IOP (PIOP). The compiler incurs a minor cost per every committed polynomial. Applying the Fiat-Shamir transformation, we obtain slightly less efficient variants of well-known PIOP-based zk-SNARKs, such as Plonk, that are knowledge-sound in the ROM under the ARSDH assumption. Importantly, there is no need for idealized group models or knowledge assumptions. This results in the first known zk-SNARKs in the ROM from falsifiable assumptions with both an efficient prover and constant-size argument.

Keywords: Black-box knowledge-soundness · polynomial commitment scheme · polynomial IOP · witness-extended emulation · zk-SNARKs

1 Introduction

Zero-knowledge Succinct Arguments of Knowledge (zk-SNARKs) allow to give a short proof of computational statements without leaking any information besides the truth of the statements. Especially in the blockchain world, efficient zk-SNARKs have found wide-scale use [BCG+14,KMS+16,Sta21] and thus are of great practical importance. Many recent zk-SNARKs are based on a combination of a polynomial commitment scheme (PCS, [KZG10]) and an information-theoretically secure non-succinct proof system like polynomial IOP [BFS20] (interactive oracle proof).

A PCS allows the prover to make a short commitment to a polynomial and later open it at a point chosen by the verifier. The very first PCS, KZG [KZG10], uses one group element for commitment and opening and two pairing operations for verification. KZG is also additively homomorphic and efficiently batchable [KZG10,TAB+20,BDFG20], making it ideal for zk-SNARKs with multiple PCS openings. On the negative side, it lacks a transparent setup, meaning that the public key cannot be generated from a public source of randomness. However, the public key is updatable, making it possible to generate it in a distributed way by different parties making sequential updates to the public key. The public key is secure if at least one honest party contributes with an update. Thus, KZG is preferred for *communication* and *verifier*-efficient updatable and universal zk-SNARKs such as [GWC19,CHM$^+$20,RZ21,CFF$^+$21,LSZ22]. Universal means that the public key of the zk-SNARK, also known as the structured reference string (SRS), can be reused for many different relations that one wants to prove (but up to some relation size bound).

As mentioned, many recent works in communication-efficient updatable and universal zk-SNARKs start by constructing an information-theoretically secure proof system in some idealized model. For example, DARK [BFS20] uses polynomial IOP, Marlin [CHM$^+$20] uses Algebraic Holographic Proof (AHP), Lunar [CFF$^+$21] and Basilisk [RZ21] use Polynomial Holographic IOP (PHP), and Plonk [GWC19] uses idealized low-degree protocols. In all such information-theoretic models is that the prover sends polynomial oracles to the verifier and the verifier can make queries to the oracles. One can transform information-theoretic proofs into succinct SNARKs using an extractable PCS. The prover commits to each polynomial oracle and opens commitments at queries chosen by the verifier. Vampire [LSZ22] is a recent exception to this paradigm, giving a direct proof in AGM.

The knowledge-soundness of PCS-based zk-SNARKs relies on PCS's extractability: given that the prover succeeds in convincing the verifier after sending a polynomial commitment and an opening, there exists an extractor that can efficiently extract a committed polynomial that is consistent with the commitment. Unfortunately, it is only known [CHM$^+$20] how to prove that KZG is extractable under knowledge assumptions or in idealized models like the generic group model (GGM) and the algebraic group model (AGM, [FKL18]). For brevity, we use the acronym IGM (idealized group model) to denote any of the GGM, the AGM, or just knowledge assumptions. Hence, the knowledge-soundness of known KZG-based zk-SNARKs relies indirectly on the IGM.

On top of that, the same zk-SNARKs use the Fiat-Shamir transform, which means that they additionally rely on another strong idealization, the random oracle model (ROM). We end up in a highly undesirable situation, where efficient zk-SNARKs used in practice depend on two different idealized models. Both models, the IGM and the ROM, are known to be uninstantiable, with several papers attacking either separately. The AGM and GGM were intensively cryptanalyzed in 2022 [Zha22,ZZK22]. Zhandry and Zhang [ZZ23] recently showed that the ROM is strictly milder heuristic than Shoup's formalization of GGM [Sho97]. Knowledge assumptions, formalized as extractable one-way functions, are known to be impossible for auxiliary input of unbounded polynomial length if a particular class of indistinguishability obfuscators exist [BCPR14].

Efficient non-updatable and non-universal (not based on PCSs) zk-SNARKs are known [GGPR13,PHGR13,Gro16] that use the IGM but not the ROM. The most efficient known updatable and universal zk-SNARKs [GKM$^+$18] that rely on an IGM and do not use the ROM are too inefficient for practice.

While non-falsifiable assumptions are needed in the standard model [GW11,CGKS23], one can obtain zk-SNARKs for NP in the ROM from falsifiable assumptions. We know two types of verifier-efficient zk-SNARKs in the ROM from falsifiable assumptions. First, the zk-SNARK of Lai and Malavolta [LM19] that uses Probabilistically Checkable Proofs (PCPs). Unfortunately, PCPs are inefficient [BCGT13] (e.g., the PCP proof length is at least $\Theta(N \log^3 N)$ with a large constant, where N is the witness size), and thus this solution only has a theoretical value. In addition, PCP-based zk-SNARKs are based on inefficient arithmetizations (mathematical representation of a relation). Second, one can combine a verifier-efficient PCS in the ROM from falsifiable assumptions with a polynomial IOP. Unfortunately, the known PCSs do not result in constant-size zk-SNARKs. For example, Dory [Lee21] and [BMM$^+$21] have an efficient prover but log-length arguments.

It is unkown if zk-SNARKs with an efficient, say $O_\lambda(N \log N)$[3], prover and $O_\lambda(1)$ argument size are possible in the ROM under falsifiable assumption. Constructing one — especially with a small constant in both $O_\lambda(\cdot)$-s — is an important open problem: it allows one to base zk-SNARKs on more secure foundations without sacrificing efficiency. Ideally, one would like to prove an already well-established zk-SNARK (such as Plonk) to be secure under weaker assumptions since an efficient arithmetization (say, R1CS or Plonk's arithmetization) and much infrastructure already accompanies it.

We propose the following two questions.

- **Theoretical:** Does a zk-SNARK with $O_\lambda(1)$ proof size and an efficient prover/verifier exist that is secure in the ROM under falsifiable assumptions?
- **Practical:** Is the KZG-based Plonk secure in the ROM under falsifiable assumptions?

We answer positively to the first question and achieve notable progress on the second.

Our Contributions. In this work, we consider non-interactive (univariate) PCSs over a field $\mathbb{F}$ where both the commitment and opening phases are non-interactive (a single message). The KZG [KZG10] is a prime example. Assume n is a degree bound on committed polynomials. Although the commitment and opening phases are non-interactive, as a whole a non-interactive PCS can be viewed as a three-round protocol:

1. the prover sends a commitment C to some polynomial $f(X) \in \mathbb{F}[X]$ of degree at most n,
2. the verifier responds with an evaluation point $\alpha \in \mathbb{F}$, and
3. the prover sends $\eta = f(\alpha)$ and an opening proof π.

[3] Here, $O_\lambda(\cdot)$ is the common "Big O" notation, but we ignore $\mathsf{poly}(\lambda)$ factors.

Note that some other PCSs, such as the one in DARK [BFS20], have an interactive opening phase. In the following, unless specified otherwise, we mean a non-interactive PCS when we write PCS.

We define computational k-special-soundness and black-box extractability for PCS by following the definitions of k-special-soundness and black-box extractability for proof systems.

More precisely, a PCS satisfies computational $(n+1)$-special-soundness if there exists an efficient extractor, such that: if an efficient adversary produces $n+1$ accepting PCS transcripts $(C, \alpha_j, \eta_j, \pi_j)$ with the same commitment C but distinct evaluation points α_j, then the extractor extracts a polynomial f of degree at most n, that is consistent with the commitment C and satisfies $f(\alpha_j) = \eta_j$ for all $j = 1, \ldots, n+1$. We prove that KZG is computationally $(n+1)$-special-sound under a new but falsifiable and standard-looking assumption ARSDH (*Adaptive Rational Strong Diffie-Hellman*). ARSDH is an adaptive variant of the known assumption RSDH (*Rational Strong Diffie-Hellman*) of González and Ráfols [GR19]. Interestingly, our special-soundness reduction uses the techniques of [TAB$^+$20] to combine $n+1$ openings to a single batch opening.

We prove that ARSDH is secure in the AGM with oblivious sampling (AGMOS) [LPS23]. AGMOS is a more realistic version of AGM that additionally allows the adversary to sample group elements without knowing their discrete logarithms. We emphasize that our special-soundness proof does not depend on the AGMOS; we use the AGMOS only as a sanity check for the *falsifiable* ARSDH assumption. We also prove that ARSDH implies the strong Diffie-Hellman assumption and thus implies the evaluation binding of KZG (it is difficult to open a polynomial commitment to two different values at the same evaluation point).

Next, we define black-box extractability for non-interactive PCS. A non-interactive PCS is black-box extractable if there exists an expected probabilistic polynomial time (PPT) black-box extractor $\mathsf{Ext}_{\mathsf{bb}}$, such that for each efficient adversary $(\mathcal{A}, \mathsf{P}^*)$, where $\mathcal{A}$ produces commitments and P^* produces openings: given a maliciously generated transcript $\mathsf{tr}_0 = (C, \alpha_0, \eta_0, \pi_0)$, for randomly sampled α_0, and an oracle access to P^*, the extractor outputs a polynomial f, such that if the PCS verifier accepts tr_0 then f agrees with tr_0 (C is a commitment of f and $f(\alpha_0) = \eta_0$). The evaluation point α_0 is sampled uniformly and independently from C from some super-polynomially large subset[4] $\mathcal{F}$ of $\mathbb{F}$.

We prove that every computationally $(n+1)$-special-sound non-interactive PCS is black-box extractable. Let ck be a commitment key. The black-box extractor $\mathsf{Ext}_{\mathsf{bb}}^{\mathsf{P}^*}(\mathsf{ck}, \mathsf{tr}_0)$ rejects if the PCS verifier V rejects tr_0. Otherwise, $\mathsf{Ext}_{\mathsf{bb}}$ invokes another extractor $\mathsf{Ext}_{\mathsf{rw}}^{\mathsf{P}^*}(\mathsf{ck}, \mathsf{tr}_0)$ that outputs n transcripts $\mathbf{tr} = (\mathsf{tr}_1, \ldots, \mathsf{tr}_n)$. $\mathsf{Ext}_{\mathsf{bb}}$ rejects if V rejects tr_j for some $j \geq 1$. Otherwise, $\mathsf{Ext}_{\mathsf{bb}}$ uses the special-soundness extractor (that exists since PCS is special sound) on input $\mathsf{tr}_0 \| \mathbf{tr}$ to extract f. One complication is that $\mathsf{Ext}_{\mathsf{rw}}$ (described in the next paragraph) is an expected PPT algorithm. To prove that $\mathsf{Ext}_{\mathsf{bb}}$ works, we need to define a special-soundness reduction $\mathcal{A}_{\mathsf{ss}}$, which internally runs $\mathsf{Ext}_{\mathsf{rw}}$. However, special-

[4] In zk-SNARKs in the literature, one can have say $\mathcal{F} = \mathbb{F}$, $\mathcal{F} = \mathbb{F}^*$, $\mathcal{F} = \mathbb{F} \setminus \mathbb{H}$ for a multiplicative subgroup of $\mathbb{H}$, etc.

soundness holds against strict PPT adversaries. To resolve this mismatch, we prove a general result that if a falsifiable security game [Nao03,GW11] (one where an efficient challenger interacts with an adversary) holds respect to any strict PPT adversary, then it also holds against any expected PPT adversary.

The most challenging part of the reduction is the rewinding extractor $\mathsf{Ext}_{\mathsf{rw}}$. $\mathsf{Ext}_{\mathsf{rw}}$ has the following goal: given that the adversary can produce a single transcript $\mathsf{tr}_0 = (C, \alpha_0, \ldots)$, accepted by the PCS verifier, $\mathsf{Ext}_{\mathsf{rw}}$ produces with an overwhelming probability n more accepting transcripts that share the same commitment but have pairwise distinct second elements α_j. For a fixed commitment C, $\mathsf{Ext}_{\mathsf{rw}}$ runs P^* with distinct random evaluation points α until it obtains n accepting transcripts. The proof that $\mathsf{Ext}_{\mathsf{rw}}$ produces a correct output with an overwhelming probability in expected PPT is technical but similar to proofs of other such extractors (especially [ACK21]).

Given the above, we can conclude that the KZG PCS is black-box extractable under the ARSDH assumption.

Compiler. Following earlier works like [CHM$^+$20,CFF$^+$21,BFS20], we present a general compiler, which combines a non-interactive PCS and a polynomial IOP [BFS20] into an interactive argument.

Polynomial IOP [BFS20] is an idealized information-theoretic proof system, where in each round of a protocol, the prover sends a polynomial oracle to the verifier, and the verifier replies with a challenge. The verifier can also query the oracles. Query points are revealed to the prover, who can use them to construct polynomial oracles of the subsequent rounds. For example, in Plonk [GWC19], several polynomials are opened at $\mathfrak{z} \leftarrow_\$ \mathbb{F}$, and one polynomial is opened at $\omega \cdot \mathfrak{z}$, where ω is a known value (a primitive root of unity). In Vampire [LSZ22], some polynomials are opened at whole (known) subgroups. Finally, the verifier either rejects or accepts the proof based on the responses from the oracles.

Bünz et al. (DARK, [BFS20]) prove that when combining a polynomial IOP with a knowledge sound PCS (the prover knows the committed polynomial), one obtains an interactive argument system for the same relation. In DARK, the opening phase is an interactive argument for proving knowledge of the committed polynomial. This is a crucial difference with our work, where commitment and opening phases are non-interactive.

Our compiler follows the execution of the polynomial IOP protocol but with the following differences. First, when the polynomial IOP prover sends a polynomial oracle f, the argument's prover sends a commitment of f, the argument's verifier responds with $\chi \leftarrow_\$ \mathcal{F}$ ($\mathcal{F}$ is some superpolynomial size set), and the prover opens the commitment at the point χ.[5] Second, when the polynomial IOP verifier wants to query one of its oracles g, it sends the query point α to the prover, which then opens the commitment of g at point α.

[5] Alternatively, one can define the following version of the KZG commitment secure in the ROM. The commitment phase is (C, η, π), where $\chi = H(C)$, and H is a random oracle. This guarantees extractability. Now, in the opening phase one can use arbitrary evaluation points α' that do not have to be uniformly random.

Our compiler's security relies on the knowledge soundness of the polynomial IOP and on the PCS's black-box extractability and evaluation binding. We prove that the compiled argument satisfies witness-extended emulation (WEE, [Lin01]). Intuitively, given an adversary that breaks witness-extended emulation of the argument, we can construct an adversary that breaks the knowledge soundness of the polynomial IOP. When the argument's prover (potentially malicious) outputs a commitment and successfully opens it at an extraction point χ, the reduction extracts the polynomial f and sends it to the polynomial IOP verifier. The evaluation binding property guarantees that the queries that the polynomial IOP verifier makes to f are consistent with the evaluations that the argument's prover outputs. Otherwise, if $f(\alpha) \neq \eta$ for some claimed evaluation η outputted by the argument's prover, we get a collision (contradicting evaluation binding).

We can use the polynomial IOP of popular zk-SNARKs like [GWC19] and Marlin [CHM$^+$20] and apply our compiler with KZG. Our compiler adds only a small overhead. First, our compiler is for polynomial IOP, so it works only for polynomial IOP variants of such zk-SNARKs. Second, we must open each committed polynomial at one more random point. This results in a public coin argument, secure under the falsifiable ARSDH assumption, with $O_\lambda(1)$ proof size, $O_\lambda(N \log N)$ prover's computation time, $O_\lambda(|\mathbb{x}| + \log N)$ verifier's computation time, and $O_\lambda(N)$ length structured reference string (SRS). Here, N denotes the circuit size representing the proven relation, and $|\mathbb{x}|$ is the statement size. Importantly, the compiled variant will be universal and updatable. Some care must be taken with zero-knowledge since we introduce additional queries to the polynomial IOPs. However, this can be adjusted by introducing additional randomness to the polynomials, which adds minor extra cost. We did not try to find a generic approach as is seems to be better to be handled it in a case-by-case basis.

To our knowledge, this is the first argument system with a constant proof size and an efficient prover and verifier in the ROM under falsifiable assumptions. After applying the Fiat-Shamir heuristic, we obtain a SNARK that is secure in the ROM under the ARSDH assumption. This answers the first question we proposed in the introduction.

As for the second question (is Plonk secure in the ROM under falsifiable assumptions?), we obtain partial success. Our compiler outputs a version of Plonk, which is less efficient by a small constant factor. For each polynomial committed by the prover, we will have an additional opening proof (1 group element) and an evaluation of the polynomial (1 field element). Note that the evaluation point χ does not add to the communication size since the prover and the verifier compute χ locally using a hash function when applying the Fiat-Shamir heuristic. Plonk's (or Marlin's) efficiency also relies on several small optimizations: batching of commitment openings and the so-called Maller's trick [GWC19]. We leave it an open question if our notion of extractability is sufficient to prove the security of such optimizations. We also leave it as an open question if it is possible to tightly reduce the soundness of our SNARK to the underlying ARSDH assumption.

Extractability in AGM(OS). The extractability of the KZG PCS in the AGM was proven in [CHM$^+$20]. They also proposed a slightly less efficient version of KZG, where a knowledge component accompanies each commitment. In that case, extractability is possible under a knowledge assumption, but the commitment length is two group elements instead of one (i.e., the extra cost is one group element). When using our extraction technique, we get purely rewinding-based extractability with the cost of one group element and one field element.

Lipmaa et al. [LPS23] recently proposed AGMOS (AGM with oblivious sampling). AGMOS is a more realistic variant of the AGM [FKL18] that gives the adversary additional power to obliviously sample group elements without knowing their discrete logarithms. Moreover, [LPS23] pointed out that researchers use KZG extractability in two different senses. In many papers (e.g., [CHM$^+$20]), KZG extractability means extracting the polynomial after both the commitment and opening phase. In other papers, it means the ability of polynomial extraction after the commitment phase only. For example, Lunar [CFF$^+$21] and Plonk [GWC19] assume that one can extract the polynomial directly after the commitment phase; however, this results [LPS23] in a spurious knowledge assumption that is secure in the AGM but insecure in the standard model.

Lipmaa et al. [LPS23] analyzed KZG in AGMOS and provided an AGMOS proof that f can be extracted from the KZG commitment C and an acceptable opening (η, π) of the commitment at any evaluation point α. In particular, α does not have to be sampled from a set of superpolynomial size (one can pick $\alpha = 0$, for example). We emphasize that AGMOS is an idealized group model. We work without using any idealized group model, but we pay by having the random evaluation requirement.

2 Preliminaries

Let λ denote the security parameter. PPT stands for probabilistic polynomial time, and DPT for deterministic polynomial time. We say *expected PPT* when referring to probabilistic Turing machines whose expected running time is bounded by a polynomial in the security parameter. All adversaries are implicitly assumed to be non-uniform. Other algorithms are uniform. By $\mathbb{F}$ we denote a finite field of prime order p; let $\mathbb{F}^* := \mathbb{F} \setminus \{0\}$. We denote by $\mathbb{F}_{\leq n}[X]$ the ring of univariate polynomials with variable X over $\mathbb{F}$ of degree $\leq n$. When a is uniformly sampled from a set A, we write $a \leftarrow_{\$} A$. A negligible function δ is a function such that, for every polynomial f, there exists an integer N_f such that, if $\lambda > N_f$, then $|\delta(\lambda)| \leq 1/f(\lambda)$. We write $\delta(\lambda) \approx_\lambda 0$, when $\delta(\lambda)$ is a negligible function. By $f(\lambda) \in \mathsf{poly}(\lambda)$, we mean that the function $f(\lambda)$ is asymptotically bounded by some polynomial.

Lagrange Interpolation. Assume n is a power of two. Let ω be the n-th primitive root of unity modulo p and let $\mathbb{H} = \langle \omega \rangle$ be the multiplicative subgroup of $\mathbb{F}$ generated by ω. (ω exists, given that $n \mid (p-1)$.)

For $j \in [1, n]$, let $\ell_j(X)$ be the *j-th Lagrange polynomial*, that is, the unique degree $n-1$ polynomial, such that $\ell_j(\omega^{j-1}) = 1$ and $\ell_i(\omega^{j-1}) = 0$ for $i \neq j$. It is well known that $\ell_j(X) = \frac{(X^n - 1)\omega^{j-1}}{n(X - \omega^{j-1})}$ for $X \neq \omega^{j-1}$.

Bilinear Groups. A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are additive cyclic (thus, abelian) groups of prime order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear pairing, and $[1]_\iota$ is a fixed generator of $\mathbb{G}_\iota$. While $[1]_\iota$ is a part of p, for the sake of clarity, we often give it as an explicit input to different algorithms. The bilinear pairing is of Type-3, that is, there is no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. We use the standard bracket notation, that is, for $\iota \in \{1, 2, T\}$ and $x \in \mathbb{Z}_p$, we write $[x]_\iota$ to denote $x[1]_\iota$. We denote $\hat{e}([x]_1, [y]_2)$ by $[x]_1 \bullet [y]_2$ and assume $[1]_T = [1]_1 \bullet [1]_2$. Thus, $[x]_1 \bullet [y]_2 = [xy]_T$ for any $x, y \in \mathbb{F}$.

Assumptions. Let $d_1(\lambda), d_2(\lambda) \in \mathsf{poly}(\lambda)$. Pgen is *(d_1, d_2)-PDL (Power Discrete Logarithm [Lip12]) secure* if for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\mathcal{A}(\mathsf{p}, [(\sigma^i)_{i=0}^{d_1}]_1, [(\sigma^i)_{i=0}^{d_2}]_2) = \sigma \middle| \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \sigma \leftarrow_{\$} \mathbb{F}^*\right] \approx_\lambda 0 \ .$$

Let $d(\lambda) \in \mathsf{poly}(\lambda)$. Pgen is *d-SDH (Strong Diffie-Hellman, [BB08])* secure in $\mathbb{G}_1$, if for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{sdh}}_{\mathsf{Pgen},1,d,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{c|c} \sigma + c \neq 0 \wedge & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \sigma \leftarrow_{\$} \mathbb{Z}_p; \\ [\varphi]_1 = \frac{1}{\sigma + c} \cdot [1]_1 & (c, [\varphi]_1) \leftarrow \mathcal{A}(\mathsf{p}, ([\sigma^i]_1)_{i=0}^d, [1, \sigma]_2) \end{array}\right] \approx_\lambda 0 \ .$$

A security game is falsifiable if expressed as an interaction between an efficient challenger and an adversary [Nao03,GW11]. In Appendix A, we show that if a falsifiable game is secure respect to any strict PPT adversary, it is also secure respect to any expected PPT adversary. We use that result in several of our reductions.

2.1 Polynomial Commitment Schemes

In a *non-interactive* (univariate) polynomial commitment scheme (PCS, [KZG10]), the prover commits to a polynomial $f \in \mathbb{F}_{\leq n}[X]$ and later opens it to $f(\alpha)$ for $\alpha \in \mathbb{F}$ chosen by the verifier. A (non-randomized) non-interactive polynomial commitment scheme [KZG10] consists of the following algorithms:

Setup $\mathsf{Pgen}(1^\lambda) \mapsto \mathsf{p}$**:** Given 1^λ, return system parameters p.

Commitment key generation $\mathsf{KGen}(\mathsf{p}, n) \mapsto (\mathsf{ck}, \mathsf{tk})$**:** Given a system parameter p and an upperbound n on the polynomial degree, return $(\mathsf{ck}, \mathsf{tk})$, where ck is the commitment key and tk is the trapdoor. We assume ck implicitly contains p. In the context of this paper, we do not use the trapdoor.

Commitment $\mathsf{Com}(\mathsf{ck}, f) \mapsto C$**:** Given a commitment key ck and a polynomial $f \in \mathbb{F}_{\leq n}[X]$, return a commitment C.

Opening $\mathsf{Open}(\mathsf{ck}, C, \alpha, f) \mapsto (\eta, \pi)$**:** Given a commitment key ck, a commitment C, an evaluation point $\alpha \in \mathbb{F}$, and a polynomial $f \in \mathbb{F}_{\leq n}[X]$, return (η, π), where $\eta \leftarrow f(\alpha)$ and π is an evaluation proof.

Verification $\mathsf{V}(\mathsf{ck}, C, \alpha, \eta, \pi) \mapsto \{0, 1\}$**:** Given a commitment key ck, a commitment C, an evaluation point α, a purported evaluation $\eta =^? f(\alpha)$, and an evaluation proof π, return 1 (accept) or 0 (reject).

Thus, in a non-interactive PCS, both the commitment and the opening are non-interactive (it only consists of a single message that can be verified non-interactively). In an *interactive PCS*, either opening or verification (or both) is an interactive protocol. Most of the known PCSs, like FRI [BBHR18], Bulletproofs [BBB$^+$18], and DARK [BFS20], are interactive. Univariate KZG [KZG10] and multivariate PST [PST13] are two well-known non-interactive PCSs.

A non-interactive PCS PC is complete, if for any λ, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $n \in \mathsf{poly}(\lambda)$, $\alpha \in \mathbb{F}$, and $f \in \mathbb{F}_{\leq n}[X]$,

$$\Pr\left[\, \mathsf{V}(\mathsf{ck}, C, \alpha, \eta, \pi) = 1 \,\middle|\, (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); (\eta, \pi) \leftarrow \mathsf{Open}(\mathsf{ck}, C, \alpha, f) \,\right] = 1.$$

Definition 1. *A non-interactive polynomial commitment scheme* PC *is* (non-black-box) extractable *for* Pgen*, if for any* $n \in \mathsf{poly}(\lambda)$*, and PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* $\mathsf{Ext}_\mathcal{A}$*, such that* $\mathsf{Adv}^{\mathrm{ext}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A},\mathsf{Ext}_\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} \mathsf{V}(\mathsf{ck}, C, \alpha, \eta, \pi) = 1 \wedge \\ \begin{pmatrix} C \neq \mathsf{Com}(f(X)) \vee \\ \deg f > n \,\vee\, f(\alpha) \neq \eta \end{pmatrix} \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ r \leftarrow \mathrm{RND}_\lambda(\mathcal{A}); (C, \alpha, \eta, \pi) \leftarrow \mathcal{A}(\mathsf{ck}; r); \\ f(X) \leftarrow \mathsf{Ext}_\mathcal{A}(\mathsf{ck}; r) \end{array}\right] \approx_\lambda 0 \ ,$$

where $r \leftarrow \mathrm{RND}_\lambda(\mathcal{A})$ *denotes sampling random coins for* $\mathcal{A}$*.*

Another common property for PCS is evaluation binding [KZG10], which does not include extractability, but disallows opening the same evaluation point to different evaluations.

Definition 2. *A polynomial commitment scheme* PC *is* evaluation binding *for* Pgen*, if for any* $n \in \mathsf{poly}(\lambda)$*, and PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{evbind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} \mathsf{V}(\mathsf{ck}, C, \alpha, \eta, \pi) = 1 \wedge \\ \mathsf{V}(\mathsf{ck}, C, \alpha, \eta', \pi') = 1 \,\wedge\, \eta \neq \eta' \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ (C, \alpha, \eta, \pi, \eta', \pi') \leftarrow \mathcal{A}(\mathsf{p}, \mathsf{ck}) \end{array}\right] \approx_\lambda 0.$$

The seminal (non-randomized) KZG [KZG10] polynomial commitment scheme is defined as follows:

$\mathsf{KZG.Pgen}(\lambda)$**:** return $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$.
$\mathsf{KZG.KGen}(\mathsf{p}, n)$**:** $\mathsf{tk} = \sigma \leftarrow_\$ \mathbb{Z}_p^*$; $\mathsf{ck} \leftarrow (\mathsf{p}, [(\sigma^i)_{i=0}^n]_1, [1, \sigma]_2)$; return $(\mathsf{ck}, \mathsf{tk})$.
$\mathsf{KZG.Com}(\mathsf{ck}, f)$**:** return $C \leftarrow [f(\sigma)]_1 = \sum_{j=0}^n f_j [\sigma^j]_1$.
$\mathsf{KZG.Open}(\mathsf{ck}, C, \alpha, f)$**:** $\eta \leftarrow f(\alpha)$; $\varphi(X) \leftarrow (f(X) - \eta)/(X - \alpha)$; $\pi \leftarrow [\varphi(\sigma)]_1$; return (η, π).

$\mathsf{KZG.V}(\mathsf{ck}, C, \alpha, \eta, \pi)$: Return 1 iff $(C - \eta[1]_1) \bullet [1]_2 = \pi \bullet [\sigma - \alpha]_2$.

KZG's security is based on the fact that $(X - \alpha) \mid (f(X) - \eta) \Leftrightarrow f(\alpha) = \eta$. It is evaluation binding under the n-SDH assumption [KZG10] and non-black-box extractable in the AGM [CHM+20] under the PDL assumption and in AGMOS under the PDL and TOFR (see Appendix C) assumptions.

2.2 Succinct Zero-Knowledge Arguments

Indexed Relation. We consider relations that depend on the pairing description $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$. An indexed relation $\mathcal{R}_\mathsf{p}$ is a relation of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$, where $\mathbb{i}$ is an index (e.g., an arithmetic circuit), $\mathbb{x}$ is a statement (e.g., a public input to the circuit) and $\mathbb{w}$ is an NP-witness (e.g., a private input to the circuit) for the language $\mathcal{L}(\mathcal{R}_\mathsf{p}) := \{(\mathbb{i}, \mathbb{x}) : (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\mathsf{p}\}$. We denote $\mathbb{I}(\mathcal{R}_\mathsf{p}) = \{\mathbb{i} : (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\mathsf{p}\}$. We also consider subrelations $\mathcal{R}_{\mathsf{p},n} \subset \mathcal{R}_\mathsf{p}$, where the size of index $\mathbb{i}$ is bounded by n.

Argument System. Groth et al. [GKM+18] introduced the notion of (preprocessing) zk-SNARKs with specializable universal structured reference string (SRS). This notion formalizes the idea that the key generation for $\mathcal{R}_{\mathsf{p},n}$ can be seen as the sequential combination of two steps. First, a probabilistic algorithm KGen generates a SRS for $\mathcal{R}_{\mathsf{p},n}$ (e.g., for satisfiability of any circuit with $\leq n$ gates) and second, a deterministic algorithm Derive specializes the universal SRS into one for a specific $\mathbb{i} \in \mathbb{I}(\mathcal{R}_{\mathsf{p},n})$ (e.g., for a fixed circuit with $\leq n$ gates).

Let (P, V) be a pair of interactive algorithms where V outputs the final message (typically either 0 or 1, unless V is malicious). We denote by $\mathsf{tr} \leftarrow \langle \mathsf{P}(x), \mathsf{V}(y) \rangle$ the protocol transcript when P gets an input x and V gets an input y. For simplicity we sometimes write $\langle \mathsf{P}(x), \mathsf{V}(y) \rangle = b$ when we compare V's last message to b.

A *succinct zero-knowledge argument system* $\mathit{\Pi} = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Derive}, \mathsf{P}, \mathsf{V})$ *with specializable universal SRS for a relation family* $(\mathsf{Pgen}, \{\mathcal{R}_{\mathsf{p},n}\}_{\mathsf{p} \in \mathrm{range}(\mathsf{Pgen}), n \in \mathbb{N}})$ consists of the following algorithms.

Setup: Given 1^λ, return system parameters $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$.

Universal SRS Generation: a probabilistic algorithm $\mathsf{KGen}(\mathsf{p}, n) \rightarrow (\mathtt{srs}, \mathtt{td}_{\mathtt{srs}})$ that takes as input public parameters p and an upper bound n on the index size, and outputs $\mathtt{srs}$ together with a trapdoor. We assume that $\mathtt{srs}$ contains p.

SRS Specialization: a deterministic algorithm $\mathsf{Derive}(\mathtt{srs}, \mathbb{i}) \rightarrow (\mathsf{ek}_\mathbb{i}, \mathsf{vk}_\mathbb{i})$ that takes as input a universal SRS $\mathtt{srs}$ and an index $\mathbb{i} \in \mathbb{I}(\mathcal{R}_{\mathsf{p},n})$, and outputs a specialized SRS $\mathtt{srs}_\mathbb{i} := (\mathsf{ek}_\mathbb{i}, \mathsf{vk}_\mathbb{i})$. Here $\mathsf{ek}_\mathbb{i}$ is for the prover, and $\mathsf{vk}_\mathbb{i}$ is for the verifier. We assume that $\mathsf{ek}_\mathbb{i}$ and $\mathsf{vk}_\mathbb{i}$ contain p.

Prover/Verifier: a pair of interactive algorithms $\langle \mathsf{P}(\mathsf{ek}_\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathsf{vk}_\mathbb{i}, \mathbb{x}) \rangle = b$, where P takes a proving key $\mathsf{ek}_\mathbb{i}$ for an index $\mathbb{i}$, a statement $\mathbb{x}$, and a witness $\mathbb{w}$, s.t. $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{p},n}$, and V takes a verification key $\mathsf{vk}_\mathbb{i}$ for an index $\mathbb{i}$ and a statement $\mathbb{x}$, and either accepts $(b = 1)$ or rejects $(b = 0)$ the argument.

Π must satisfy the following four requirements.
Completeness. For all $\mathsf{p} \in \text{range}(\mathsf{Pgen})$, $n \in \mathbb{N}$, $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{p},n}$,

$$\Pr\left[\langle \mathsf{P}(\mathsf{ek}_{\mathbb{i}}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x})\rangle = 1 \,\middle|\, \begin{array}{l} (\mathtt{srs}, \mathsf{td}_{\mathtt{srs}}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ (\mathsf{ek}_{\mathbb{i}}, \mathsf{vk}_{\mathbb{i}}) \leftarrow \mathsf{Derive}(\mathtt{srs}, \mathbb{i}) \end{array}\right] = 1 \ .$$

Witness-Extended Emulation. Π satisfies witness-extended emulation if for every DPT P^* there exists an expected polynomial time emulator Emu, such that for any λ, PPT adversary $\mathcal{A}$, PPT distinguisher $\mathcal{D}$, and $n \in \mathsf{poly}(\lambda)$, $\mathsf{Adv}^{\mathrm{wee}}_{\mathsf{Pgen},\Pi,n,\mathsf{P}^*,\mathcal{A},\mathcal{D}}(\lambda) := |\varepsilon_0 - \varepsilon_1| \approx_\lambda 0$, where

$$\varepsilon_0 := \Pr\left[\begin{array}{l} \mathbb{i} \in \mathbb{I}(\mathcal{R}_{\mathsf{p},n}) \wedge \\ \mathcal{D}(\mathsf{tr}) = 1 \end{array} \,\middle|\, \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathtt{srs}, \mathsf{td}_{\mathtt{srs}}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ (\mathbb{i}, \mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{p}, \mathtt{srs}); (\mathsf{ek}_{\mathbb{i}}, \mathsf{vk}_{\mathbb{i}}) \leftarrow \mathsf{Derive}(\mathtt{srs}, \mathbb{i}); \\ \mathsf{tr} \leftarrow \langle \mathsf{P}^*(\mathsf{p}, \mathtt{srs}, \mathbb{x}, \mathsf{st}), \mathsf{V}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x})\rangle \end{array}\right] ,$$

$$\varepsilon_1 := \Pr\left[\begin{array}{l} \mathbb{i} \in \mathbb{I}(\mathcal{R}_{\mathsf{p},n}) \wedge \\ \mathcal{D}(\mathsf{tr}) = 1 \wedge \\ \begin{pmatrix} \mathsf{V}_{\mathsf{check}}(\mathtt{srs}, \mathsf{tr}) = 1 \\ \Rightarrow (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R} \end{pmatrix} \end{array} \,\middle|\, \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathtt{srs}, \mathsf{td}_{\mathtt{srs}}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ (\mathbb{i}, \mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{p}, \mathtt{srs}); \\ (\mathsf{ek}_{\mathbb{i}}, \mathsf{vk}_{\mathbb{i}}) \leftarrow \mathsf{Derive}(\mathtt{srs}, \mathbb{i}); \\ (\mathsf{tr}, \mathbb{w}) \leftarrow \mathsf{Emu}^{\langle \mathsf{P}^*(\mathsf{p}, \mathtt{srs}, \mathbb{x}, \mathsf{st}), \mathsf{V}(\mathsf{vk}_{\mathcal{R}}, \mathbb{x})\rangle}(\mathsf{p}, \mathtt{srs}, \mathbb{x}) \end{array}\right] ,$$

where Emu has access to a transcript oracle that can be rewound to any round and run again with fresh random coins of the verifier. $\mathsf{V}_{\mathsf{check}}(\mathtt{srs}, \mathsf{tr})$ outputs 1 if the transcript is accepted by the verifier and 0 otherwise.
Honest Verifier Zero-Knowledge. Π is ε-statistical *honest verifier zero-knowledge* if there exists a PPT simulator Sim, s.t. for all unbounded algorithms $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$, all $\mathsf{p} \in \text{range}(\mathsf{Pgen})$, all $n \in \mathsf{poly}(\lambda)$, $|\varepsilon_0(\lambda) - \varepsilon_1(\lambda)| \leq \varepsilon(\lambda)$, where

$$\varepsilon_0(\lambda) := \Pr\left[\begin{array}{l} \mathcal{D}_2(\mathsf{st}, \mathsf{tr}) = 1 \wedge \\ \mathcal{R}_{\mathsf{p},n}(\mathbb{i}, \mathbb{x}, \mathbb{w}) \end{array} \,\middle|\, \begin{array}{l} (\mathtt{srs}, \mathsf{td}_{\mathtt{srs}}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); (\mathbb{i}, \mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow \mathcal{D}_1(\mathtt{srs}); \\ (\mathsf{ek}_{\mathbb{i}}, \mathsf{vk}_{\mathbb{i}}) \leftarrow \mathsf{Derive}(\mathtt{srs}, \mathbb{i}); \\ \mathsf{tr} \leftarrow \langle \mathsf{P}\left(\mathsf{ek}_{\mathbb{i}}, \mathbb{x}, \mathbb{w}\right), \mathsf{V}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x})\rangle \end{array}\right] ,$$

$$\varepsilon_1(\lambda) := \Pr\left[\begin{array}{l} \mathcal{D}_2(\mathsf{st}, \mathsf{tr}) = 1 \wedge \\ \mathcal{R}_{\mathsf{p},n}(\mathbb{i}, \mathbb{x}, \mathbb{w}) \end{array} \,\middle|\, \begin{array}{l} (\mathtt{srs}, \mathsf{td}_{\mathtt{srs}}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); (\mathbb{i}, \mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow \mathcal{D}_1(\mathtt{srs}); \\ (\mathsf{ek}_{\mathbb{i}}, \mathsf{vk}_{\mathbb{i}}) \leftarrow \mathsf{Derive}(\mathtt{srs}, \mathbb{i}); \\ \mathsf{tr} \leftarrow \langle \mathsf{Sim}\left(\mathtt{srs}, \mathsf{td}_{\mathtt{srs}}, \mathbb{i}, \mathbb{x}\right), \mathsf{V}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x})\rangle \end{array}\right] .$$

We say that Π has statistical honest verifier zero-knowledge when $\varepsilon(\lambda)$ is negligible and perfect zero-knowledge when $\varepsilon(\lambda) = 0$.
Succinctness. Π is *succinct* if the running time of V is $\mathsf{poly}(\lambda + |\mathbb{x}| + \log |\mathbb{w}|)$ and the communication size is $\mathsf{poly}(\lambda + \log |\mathbb{w}|)$.

Π is *updatable* [GKM+18], if the SRS can be sequentially updated by many updaters, such that knowledge-soundness holds if either the original SRS creator or one of the updaters is honest.

When we have a public-coin protocol with a constant number of rounds, we can apply the Fiat-Shamir heuristic [FS87] to obtain a zk-SNARK.

3 ARSDH: Underlying Security Assumption

For a set $\mathcal{S}$, $\mathbf{Z}_{\mathcal{S}}(X) := \prod_{s \in \mathcal{S}}(X - s)$ is its vanishing polynomial. We need a new assumption, ARSDH, that is an adaptive version of the following known assumption.

Definition 3 (RSDH **[GR19]).** *Let $n \in \mathsf{poly}(\lambda)$ and* Pgen *be a bilinear-group generator. Let $\mathcal{S} = \{\alpha_j\} \subset \mathbb{F}$ be any set of size $n+1$. Then, the $\mathcal{S}$-*RSDH *(*Rational Strong Diffie-Hellman*) assumption holds for* Pgen *in $\mathbb{G}_1$, if for any PPT $\mathcal{A}$, the following probability is negligible:* $\mathsf{Adv}^{\mathsf{rsdh}}_{\mathsf{Pgen},1,n,\mathcal{S},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l}[g]_1 \neq [0]_1 \wedge \\ [g]_1 \bullet [1]_2 = [\varphi]_1 \bullet [\mathbf{Z}_{\mathcal{S}}(\sigma)]_2\end{array}\middle| \begin{array}{c}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \sigma \leftarrow_{\$} \mathbb{Z}_p; \\ \mathsf{ck} \leftarrow ([(\sigma^i)_{i=0}^{n}]_1, [(\sigma^i)_{i=0}^{n+1}]_2); \\ [g, \varphi]_1 \leftarrow \mathcal{A}(\mathsf{ck}, \mathcal{S})\end{array}\right] .$$

(The condition $[g]_1 \bullet [1]_2 = [\varphi]_1 \bullet [\mathbf{Z}_{\mathcal{S}}(\sigma)]_2$ is equivalent to $\frac{1}{\mathbf{Z}_{\mathcal{S}}(\sigma)} \cdot [g]_1 = [\varphi]_1$.)

Definition 4 (New Assumption ARSDH**).** *We say the* adaptive $(n+1)$-RSDH *assumption holds for* Pgen *in $\mathbb{G}_1$ if $\mathcal{S}$-*RSDH *holds (with slightly different input to $\mathcal{A}$) for* Pgen *in $\mathbb{G}_1$ even when the adversary can choose itself a set $\mathcal{S}$ of size $n+1$. That is, if for any PPT $\mathcal{A}$, the following probability is negligible:* $\mathsf{Adv}^{\mathsf{arsdh}}_{\mathsf{Pgen},1,n,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l}\mathcal{S} \subset \mathbb{F} \wedge |\mathcal{S}| = n+1 \wedge [g]_1 \neq [0]_1 \wedge \\ [g]_1 \bullet [1]_2 = [\varphi]_1 \bullet [\mathbf{Z}_{\mathcal{S}}(\sigma)]_2\end{array}\middle| \begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \sigma \leftarrow_{\$} \mathbb{Z}_p; \\ \mathsf{ck} \leftarrow ([(\sigma^i)_{i=0}^{n}]_1, [1, \sigma]_2); \\ (\mathcal{S}, [g, \varphi]_1) \leftarrow \mathcal{A}(\mathsf{ck})\end{array}\right] .$$

The ($\mathcal{S}$-)RSDH assumption from [GR19] is stronger than ARSDH in the sense that it gives the adversary $[(\sigma^i)_{i=0}^{n}]_1, [(\sigma^i)_{i=0}^{n+1}]_2$ as an input. The additional input elements were needed for the assumption to be publicly verifiable, meaning breaking the success of the adversary can be tested only by knowing ck (knowing σ itself is unnecessary). In our application, public verification is not important. On the other hand, [GR19] assumed $\mathcal{S}$-RSDH (for a specific set $\mathcal{S}$) while ARSDH must be secure against an adverary that adaptively chooses $\mathcal{S}$.

In our constructions, we need both the ARSDH and SDH assumptions. The latter is needed for the evaluation binding of the KZG commitment. To simplify the assumption zoo, we prove the following lemma. Note that the adaptive choice of $\mathcal{S}$ in ARSDH is needed for the reduction to work (in addition to being needed in Section 4.1).

Lemma 1. $(n+1)$-ARSDH *implies* $(n+1)$-SDH.

Proof. Let $\mathcal{A}$ be an $(n+1)$-SDH adversary. We construct the following $(n+1)$-ARSDH adversary $\mathcal{B}(\mathsf{ck})$:

1. Obtain $(c, [\varphi]_1 = \frac{1}{\sigma+c}[1]_1) \leftarrow \mathcal{A}(\mathsf{ck})$.
2. Abort if $\mathcal{A}(\mathsf{ck})$ did not succeed, i.e, if $[\varphi]_1 \bullet [\sigma + c]_2 \neq [1]_T$ or $\sigma + c = 0$.
3. Choose *any* set $\mathcal{S}$ of size $n+1$ that contains $-c$, but not σ.
4. Set $g(X) \leftarrow \mathbf{Z}_{\mathcal{S}}(X)/(X+c) \in \mathbb{F}[X]$, with degree n.
5. Return $(\mathcal{S}, [g(\sigma), \varphi]_1)$.

Clearly, $\mathcal{B}$ has broken RSDH since $\frac{1}{\mathbf{Z}_{\mathcal{S}}(\sigma)}[g(\sigma)]_1 = [\varphi]_1$ and $g(\sigma) \neq 0$. □

In particular, this means that ARSDH implies that KZG is evaluation binding. In Appendix B, we prove the security of ARSDH in the algebraic group model with oblivious sampling (AGMOS) [LPS23]. This is the recent variant of AGM [FKL18], which additionally allows oblivious sampling of group elements. It does not mean that the security of our protocols relies on AGM/AGMOS. Instead, we use AGMOS only as an additional sanity check.

4 Special Soundness of KZG

In the following sections, we define two security notions for non-interactive polynomial commitment schemes: special soundness and black-box extractability (BBE). We prove that KZG satisfies both notions under ARSDH.

We model a non-interactive polynomial commitment scheme (e.g., KZG) as a three-message protocol, where the first message is a commitment, the second is the verifier's query (evaluation point) α, and the third is an evaluation η together with an evaluation proof π.

Notation. We call $\mathsf{tr} := (C, \alpha, \eta, \pi)$ a *transcript.* For a fixed ck, tr is *accepting* if $\mathsf{V}(\mathsf{ck}, \mathsf{tr}) = 1$. Let $n \geq 1$ be an integer. We say that an $(n+1)$-tuple $\mathbf{tr} = \{\mathsf{tr}_j = (C_j, \alpha_j, \eta_j, \pi_j) : j \in [0, n]\}$ is *admissible*, if (1) $C_i = C_j =: C$ for all $i, j \in [0, n]$, and (2) $\alpha_i \neq \alpha_j$ for $i \neq j$. We say $\mathbf{tr}$ is *accepting* if each tr_j is accepting.

For a non-interactive PCS PC, $\lambda \in \mathbb{N}$, $n \in \mathsf{poly}(\lambda)$, $\mathsf{p} \in \mathsf{Pgen}(1^\lambda)$, $\mathsf{ck} \in \mathsf{PC.KGen}(\mathsf{p}, n)$ (that encodes implicitly information about PC, λ, n, and p), and an admissible $(n+1)$-tuple $\mathbf{tr}$, we define the following two relations:

$$\begin{aligned} \mathcal{R}_{\mathsf{ck}} &:= \{(C, f) : C = \mathsf{PC.Com}(\mathsf{ck}, f) \wedge \deg f \leq n\} \ , \\ \mathcal{R}_{\mathsf{ck},\mathbf{tr}} &:= \{(C, f) : (C, f) \in \mathcal{R}_{\mathsf{ck}} \wedge \forall j \in [0, n]. f(\alpha_j) = \eta_j\} \ . \end{aligned} \tag{1}$$

That is, (C, f) belongs to $\mathcal{R}_{\mathsf{ck}}$ if $f(X)$ is a valid opening of C. Moreover, (C, f) belongs to $\mathcal{R}_{\mathsf{ck},\mathbf{tr}}$ if it belongs to $\mathcal{R}_{\mathsf{ck}}$ and in addition, $f(\alpha_j) = \eta_j$ for all $j \in [0, n]$.

4.1 Special Soundness

Next, we define a variant of the standard special soundness [CDS94] notion for *non-interactive* polynomial commitment schemes.

Definition 5 (Special Soundness). *Let $n \in \mathsf{poly}(\lambda)$ with $n \geq 1$. A non-interactive polynomial commitment scheme PC is $\underline{\text{computationally } (n+1)\text{-special-sound}}$ for Pgen, if there exists a PPT extractor $\mathsf{Ext}_{\mathsf{ss}}$, such that for any PPT adversary $\mathcal{A}_{\mathsf{ss}}$, $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n+1,\mathcal{A}_{\mathsf{ss}}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l|l} \mathbf{tr} = (\mathsf{tr}_j)_{j=0}^{n} \wedge & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \\ \forall j \in [0, n]. \begin{pmatrix} \mathsf{tr}_j = (C, \alpha_j, \eta_j, \pi_j) \\ \wedge \mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 1 \end{pmatrix} & (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ & \mathbf{tr} \leftarrow \mathcal{A}_{\mathsf{ss}}(\mathsf{ck}); \\ \wedge\ (\forall i \neq j. \alpha_i \neq \alpha_j) \wedge (C, f) \notin \mathcal{R}_{\mathsf{ck},\mathbf{tr}} & f \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}) \end{array}\right] \approx_\lambda 0 \ .$$

Intuitively, this definition states that if $\mathcal{A}_{\mathsf{ss}}$ produces an accepting admissible $(n+1)$-tuple $\mathbf{tr}$, then one can extract a degree-$\leq n$ polynomial $f(X)$ that agrees with all the transcripts (i.e., with all $n+1$ polynomial openings).

Theorem 1. *If the $(n+1)$-ARSDH assumption holds, then KZG for degree $\leq n$ polynomials is computationally $(n+1)$-special-sound: There exists a DPT extractor $\mathsf{Ext}_{\mathsf{ss}}$, such that for any PPT $\mathcal{A}_{\mathsf{ss}}$, there exists a PPT $\mathcal{B}$, such that* $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n+1,\mathcal{A}_{\mathsf{ss}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{arsdh}}_{\mathsf{Pgen},1,n+1,\mathcal{B}}(\lambda)$.

We fix some notation and state a technical lemma before proving Theorem 1. Let $\mathcal{I} := [0,n]$ and fix any set $\mathcal{S} := \{\alpha_j\}_{j\in\mathcal{I}}$ with $\alpha_i \neq \alpha_j$. As before, let $\mathbf{Z}_{\mathcal{S}}(X) := \prod_{j\in\mathcal{I}}(X-\alpha_j)$ be the vanishing polynomial of $\mathcal{S}$. For $j \in \mathcal{I}$, let

$$\ell_j^{\mathcal{I}}(X) := \prod_{i\neq j\in\mathcal{I}} \frac{X-\alpha_i}{\alpha_j-\alpha_i}$$

be the jth Lagrange polynomial of $\mathcal{S}$ over $\mathcal{I}$. Let

$$\mathsf{d}_j^{\mathcal{I}} := \frac{1}{\mathbf{Z}_{\mathcal{S}\setminus\{\alpha_j\}}(\alpha_j)} = \frac{1}{\prod_{i\neq j\in\mathcal{I}}(\alpha_j-\alpha_i)} \enspace .$$

Clearly,

$$\ell_j^{\mathcal{I}}(X) = \frac{\mathbf{Z}_{\mathcal{S}}(X)\mathsf{d}_j^{\mathcal{I}}}{X-\alpha_j} \enspace . \tag{2}$$

In Lemma 2, we generalize a batching technique of Tomescu et al. [TAB+20] from the case $\alpha_j \in \langle\omega\rangle$ to any $\alpha_j \in \mathbb{F}$.[6]

Lemma 2 (Batching lemma). *Let $\mathcal{S} = \{\alpha_j\}_{j\in\mathcal{I}}$ with $\alpha_i \neq \alpha_j$. Assume that for all $j \in \mathcal{I}$,*

$$[\mathsf{c}-\eta_j]_1 \bullet [1]_2 = [\varphi_j]_1 \bullet [\sigma-\alpha_j]_2 \tag{3}$$

for some $[\mathsf{c}]_1$, $[\varphi_j]_1$, and η_j. Then,

$$[\mathsf{c}-L(\sigma)]_1 \bullet [1]_2 = [\varphi]_1 \bullet [\mathbf{Z}_{\mathcal{S}}(\sigma)]_2 \enspace , \tag{4}$$

where $[\varphi]_1 := \sum_{j\in\mathcal{I}} \mathsf{d}_j^{\mathcal{I}}[\varphi_j]_1$ and $L(X) := \sum_{j\in\mathcal{I}} \eta_j \ell_j^{\mathcal{I}}(X)$ is the low-degree extension of $\{\eta_j\}$.

Proof. Let us first handle the case $\sigma = \alpha_{j_0}$ for some j_0. In this case, Eq. (3) tells us that $\mathsf{c} = \eta_{j_0}$. Moreover, $\ell_j^{\mathcal{I}}(\sigma) = 0$ for $j \neq j_0$ while

$$\ell_{j_0}^{\mathcal{I}}(X) = \mathsf{d}_{j_0}^{\mathcal{I}} \prod_{i\neq j_0}(X-\alpha_i) = \frac{\mathbf{Z}_{\mathcal{S}\setminus\{\sigma\}}(X)}{\mathbf{Z}_{\mathcal{S}\setminus\{\sigma\}}(\sigma)} \enspace .$$

Since $\mathsf{d}_{j_0}^{\mathcal{I}} = 1/\mathbf{Z}_{\mathcal{S}\setminus\{\sigma\}}(\sigma)$, we get that $L(\sigma) = \sum_{j\in\mathcal{I}} \eta_j \ell_j^{\mathcal{I}}(\sigma) = \eta_{j_0} = \mathsf{c}$ and Eq. (4) holds.

[6] [TAB+20] used this technique of batching polynomial commitment openings to improve on efficiency, while we use it for the security proof. Without using Lemma 2, we have a somewhat uglier assumption, where $\mathcal{A}_{\mathsf{ss}}$ returns $[\varphi_j]_1$ for every $j \in \mathcal{I}$, and $n+1$ equalities $[g]_1 \bullet [1]_2 = [\varphi_j]_1 \bullet [\sigma-\alpha_j]_2$ hold individually.

```
$\mathsf{Ext_{ss}}(\mathsf{ck} = ([(\sigma^i)_{i=0}^n]_1, [1, \sigma]_2), \mathbf{tr})$
    if $\exists i \neq j . \alpha_i = \alpha_j$ then return $\perp$;
    if $\exists j \in \mathcal{I} : \mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 0$ then
        return $\perp$;
    $L(X) \leftarrow \sum_{j \in \mathcal{I}} \eta_j \ell_j^{\mathcal{I}}(X) \in \mathbb{F}_{\leq n}[X]$;
    $[g]_1 \leftarrow [\mathsf{c} - L(\sigma)]_1$;
    if $[g]_1 = [0]_1$ then return $L(X)$;
    return $\perp$;
```

```
$\mathcal{B}(\mathsf{ck} = ([(\sigma^i)_{i=0}^n]_1, [1, \sigma]_2))$
    $\mathbf{tr} \leftarrow \mathcal{A}_{\mathsf{ss}}(\mathsf{ck})$;   // $\mathsf{tr}_j = ([\mathsf{c}]_1, \alpha_j, \eta_j, [\varphi_j]_1)$
    if $\exists i \neq j . \alpha_i = \alpha_j$ then return $\perp$;
    if $\exists j \in \mathcal{I} : \mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 0$ then
        return $\perp$;
    $L(X) \leftarrow \sum_{j \in \mathcal{I}} \eta_j \ell_j^{\mathcal{I}}(X) \in \mathbb{F}_{\leq n}[X]$;
    $[g]_1 \leftarrow [\mathsf{c} - L(\sigma)]_1$;
    if $[g]_1 = [0]_1$ then return $\perp$;
    $[\varphi]_1 \leftarrow \sum_{j \in \mathcal{I}} \mathsf{d}_j^{\mathcal{I}} [\varphi_j]_1$;
    return $(\mathcal{S} \leftarrow \{\alpha_j\}, [g, \varphi]_1)$;
```

Fig. 1. The extractor $\mathsf{Ext_{ss}}$ and the ARSDH reduction $\mathcal{B}$ in the proof of Theorem 1

From now on, assume $\sigma \notin \mathcal{S}$. Define implicitly $\varphi := (\mathsf{c} - L(\sigma))/\mathbf{Z}_{\mathcal{S}}(\sigma) = \mathsf{c}/\mathbf{Z}_{\mathcal{S}}(\sigma) - L(\sigma)/\mathbf{Z}_{\mathcal{S}}(\sigma)$. From Eq. (2), we get

$$\frac{L(\sigma)}{\mathbf{Z}_{\mathcal{S}}(\sigma)} = \frac{\sum_{j \in \mathcal{I}} \eta_j \ell_j^{\mathcal{I}}(\sigma)}{\mathbf{Z}_{\mathcal{S}}(\sigma)} = \frac{\sum_{j \in \mathcal{I}} \eta_j \mathbf{Z}_{\mathcal{S}}(\sigma) \mathsf{d}_j^{\mathcal{I}} / (\sigma - \alpha_j)}{\mathbf{Z}_{\mathcal{S}}(\sigma)} = \sum_{j \in \mathcal{I}} \frac{\mathsf{d}_j^{\mathcal{I}} \eta_j}{\sigma - \alpha_j} \quad .$$

We also get from Eq. (2) that $\frac{1}{\mathbf{Z}_{\mathcal{S}}(X)} = \frac{1}{\mathbf{Z}_{\mathcal{S}}(X)} \sum_{j \in \mathcal{I}} \ell_j^{\mathcal{I}}(X) = \sum_{j \in \mathcal{I}} \frac{\mathsf{d}_j^{\mathcal{I}}}{X - \alpha_j}$. Thus, $\mathsf{c}/\mathbf{Z}_{\mathcal{S}}(\sigma) = \sum_{j \in \mathcal{I}} \mathsf{d}_j^{\mathcal{I}} \mathsf{c}/(\sigma - \alpha_j)$. Hence,

$$\varphi = \sum_{j \in \mathcal{I}} \frac{\mathsf{d}_j^{\mathcal{I}} \mathsf{c}}{\sigma - \alpha_j} - \sum_{j \in \mathcal{I}} \frac{\mathsf{d}_j^{\mathcal{I}} \eta_j}{\sigma - \alpha_j} = \sum_{j \in \mathcal{I}} \mathsf{d}_j^{\mathcal{I}} \frac{\mathsf{c} - \eta_j}{\sigma - \alpha_j} = \sum_{j \in \mathcal{I}} \mathsf{d}_j^{\mathcal{I}} \varphi_j \quad .$$

The last equality holds since $\varphi_j = (\mathsf{c} - \eta_j)/(\sigma - \alpha_j)$. This proves the lemma. □

Proof (Of Theorem 1). Let $\mathcal{A}_{\mathsf{ss}}$ be any PPT adversary in the computational special soundness game. We construct the following extractor $\mathsf{Ext_{ss}}$. (See Fig. 1 for a formal description.)

Let $n \in \mathsf{poly}(\lambda)$, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, and $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n)$. $\mathsf{Ext_{ss}}$ obtains ck and $\mathbf{tr}$, where $\mathbf{tr}$ is an $(n+1)$-tuple of transcripts $\mathsf{tr}_j = ([\mathsf{c}]_1, \alpha_j, \eta_j, [\varphi_j]_1)$. Define $\mathcal{S} := \{\alpha_j\}_{j \in \mathcal{I}}$.[7] When $\mathcal{A}_{\mathsf{ss}}$ produces a successful attack, α_j are pairwise different ($\mathbf{tr}$ is admissible), and the KZG verifier accepts each transcript. That is, for all $j \in \mathcal{I}$, Eq. (3) holds.

$\mathsf{Ext_{ss}}$ interpolates a polynomial $L(X)$ of degree $\leq n$ such that $L(\alpha_j) = \eta_j$ for every $j \in \mathcal{I}$. That is, $L(X) = \sum_{j \in \mathcal{I}} \eta_j \ell_j^{\mathcal{I}}(X)$. If $[\mathsf{c}]_1 = [L(\sigma)]_1$, $\mathsf{Ext_{ss}}$ outputs $L(X)$. Observe that in this case, $\deg L(X) \leq n$ and $L(\alpha_j) = \eta_j$ for all $j \in \mathcal{I}$; thus, $(C, L) \in \mathcal{R}_{\mathsf{ck},\mathbf{tr}}$ as required in Definition 5. Otherwise, $\mathsf{Ext_{ss}}$ outputs $\perp$.

Let bad be the event that $\mathsf{c} \neq L(\sigma)$ but the verifier accepts all transcripts in $\mathbf{tr}$. In Fig. 1, we depict a reduction $\mathcal{B}$ that breaks ARSDH whenever bad happens.

[7] Note that if $\mathcal{S}$ contains σ then $\mathcal{A}_{\mathsf{ss}}$ has broken the $(n, 1)$-PDL assumption, and thus also the ARSDH assumption. However, the following proof also goes through when $\mathcal{S}$ contains σ, and thus we do not have to consider the case $\sigma \in \mathcal{S}$ separately.

$\mathcal{B}$ runs $\mathcal{A}_{\mathsf{ss}}$ to obtain transcripts and then computes $L(X)$ just as the extractor. If $[\mathsf{c}]_1 = [L(\sigma)]_1$, $\mathcal{B}$ outputs $\perp$ (in this case, the extractor succeeds). Otherwise, $\mathcal{B}$ proceeds and computes $[g]_1 \leftarrow [\mathsf{c} - L(\sigma)]_1$ and $[\varphi]_1 \leftarrow \sum_{j \in \mathcal{I}} \mathsf{d}_j^{\mathcal{I}} [\varphi_j]_1$. Since Eq. (3) holds for all $j \in \mathcal{I}$, we get from Lemma 2 that Eq. (4) holds, that is, $[g]_1 \bullet [1]_2 = [\varphi]_1 \bullet [\mathbf{Z}_{\mathcal{S}}(\sigma)]_2$. When bad happens, $[g]_1 \neq [0]_1$ and thus $\mathcal{B}$ breaks the ARSDH assumption by returning $[g, \varphi]_1$. Thus, $\Pr[\mathcal{B}$ breaks $\mathsf{ARSDH}] = \Pr[\mathsf{bad}]$. Summarizing, $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n+1,\mathcal{A}_{\mathsf{ss}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{arsdh}}_{\mathsf{Pgen},1,n+1,\mathcal{B}}(\lambda)$. $\square$

5 Rewinding Lemma

Next, we prove a generic information-theoretically secure rewinding lemma. Intuitively, if the adversary can produce an accepting transcript for a random challenge α_0, then there exists an efficient extractor that can recover n more accepting transcripts for $n \in \mathsf{poly}(\lambda)$. Note that $\varepsilon_{\mathsf{ss}}$ is a function of p since $|\mathcal{F}_{\mathsf{p}}|$ is a function of p.

Theorem 2. *Fix $\mathcal{F}_{\mathsf{p}}$ as a function of p. For all DPT P^* and $n \in \mathsf{poly}(\lambda)$, there exists an expected PT extractor $\mathsf{Ext}_{\mathsf{rw}}$, such that for any unbounded $\mathcal{A}$, $\varepsilon_{\mathsf{ss}}(\mathsf{p}) > 1 - n/|\mathcal{F}_{\mathsf{p}}|$, where for every $\mathsf{p} \in \mathsf{Pgen}(1^\lambda)$, $\varepsilon_{\mathsf{ss}}(\mathsf{p}) :=$*

$$\Pr\left[\begin{array}{c|l} \mathsf{V}(\mathsf{ck}, \mathsf{tr}_0) = 1 \Rightarrow & (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); (C, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{ck}); \\ \quad \forall j \in [1, n]. \mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 1 & \alpha_0 \leftarrow_{\$} \mathcal{F}_{\mathsf{p}}; (\eta_0, \pi_0) \leftarrow \mathsf{P}^*(\mathsf{st}, \alpha_0); \\ & \mathsf{tr}_0 \leftarrow (C, \alpha_0, \eta_0, \pi_0); \\ & \mathbf{tr} \leftarrow \mathsf{Ext}_{\mathsf{rw}}^{\mathsf{P}^*(\mathsf{st}, \cdot)}(\mathsf{ck}, \mathsf{tr}_0) \end{array}\right],$$

and $\mathbf{tr} = (\mathsf{tr}_1, \ldots, \mathsf{tr}_n)$ consists of transcripts $\mathsf{tr}_j = (C, \alpha_j, \eta_j, \pi_j)$ for $j \in [1, n]$, where C is the same as in tr_0 and $\alpha_j \in \mathcal{F}_{\mathsf{p}}$ are pairwise distinct. In particular, $\mathsf{Ext}_{\mathsf{rw}}$ makes an expected number of n queries to P^*.

Proof. Our proof strategy roughly follows [ACK21]. In Fig. 2, we depict the extractor $\mathsf{Ext}_{\mathsf{rw}}$. Intuitively, $\mathsf{Ext}_{\mathsf{rw}}$ runs $\mathsf{P}^*(\mathsf{st}, \cdot)$ on distinct uniformly random challenges $\alpha \in \mathcal{F}_{\mathsf{p}}$ until it either finds n additional accepting transcripts or the whole challenge set $\mathcal{F}_{\mathsf{p}}$ is exhausted.

Let r_{p}, r_{ck}, and $r_{\mathcal{A}}$ be the randomizers used in creating $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda; r_{\mathsf{p}})$, $\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{p}, n; r_{\mathsf{ck}})$, and $(C, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{ck}; r_{\mathcal{A}})$. Consider a Boolean matrix H where the rows are indexed by the set

$$\mathsf{Rows} := \{\bar{r} = (r_{\mathsf{p}}, r_{\mathsf{ck}}, r_{\mathcal{A}}) : r_{\mathsf{p}}, r_{\mathsf{ck}}, r_{\mathcal{A}} \in \{0,1\}^{\mathsf{poly}(\lambda)}\}$$

and the columns are indexed by the verifier challenges from $\mathcal{F}_{\mathsf{p}}$. We implicitly set $H_{\bar{r},\alpha} = 1$ iff $\mathsf{V}(\mathsf{ck}, C, \alpha, \eta, \pi) = 1$, where the parameters $(\mathsf{p}, \mathsf{ck}, C, \mathsf{st})$ are created using the coins $\bar{r}$ and $(\eta, \pi) \leftarrow \mathsf{P}^*(\mathsf{st}, \alpha)$.

Probability analysis. Consider the game described in the lemma's statement. W.r.t. this game, let A be the event that $\mathsf{V}(\mathsf{ck}, \mathsf{tr}_0) = 1$ and let B be the event that $\forall j \in [1, n]$, $\mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 1$. Then,

$$\varepsilon_{\mathsf{ss}}(\mathsf{p}) = \Pr[A \Rightarrow B] = \Pr[A \wedge (A \Rightarrow B)] + \Pr[\neg A \wedge (A \Rightarrow B)]$$

```
$\mathsf{Ext}_{\mathsf{rw}}^{\mathsf{P}^*(\mathsf{st},\cdot)}(\mathsf{ck}, \mathsf{tr}_0)$
Parse $\mathsf{tr}_0 = (C, \alpha_0, \eta_0, \pi_0)$;
if $\mathsf{V}(\mathsf{ck}, C, \alpha_0, \eta_0, \pi_0) = 0$ then return $\perp$; fi
$j \leftarrow 1; \mathcal{T} \leftarrow \mathcal{F}_{\mathsf{p}} \setminus \{\alpha_0\}$;
while $j \leq n \wedge \mathcal{T} \neq \emptyset$ do
  $\alpha \leftarrow_{\$} \mathcal{T}; (\eta, \pi) \leftarrow \mathsf{P}^*(\mathsf{st}, \alpha)$;
  if $\mathsf{V}(\mathsf{ck}, C, \alpha, \eta, \pi) = 1$ then $\mathsf{tr}_j \leftarrow (C, \alpha, \eta, \pi); j \leftarrow j + 1$; fi
  $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\alpha\}$;
endwhile
if $j < n$ then return $\perp$; fi
return $\mathbf{tr} \leftarrow (\mathsf{tr}_1, \ldots, \mathsf{tr}_n)$;
```

Fig. 2. Extractor $\mathsf{Ext}_{\mathsf{rw}}$ from Theorem 2.

$$= \Pr[A \wedge B] + \Pr[\neg A] \ .$$

Let $R := |\mathsf{Rows}|$. For $j \leq |\mathcal{F}_{\mathsf{p}}|$, let R_j be the number of rows in H with exactly j ones. Thus, $\Pr[A] = \left(\sum_{j=1}^{|\mathcal{F}_{\mathsf{p}}|} jR_j\right) / (R \cdot |\mathcal{F}_{\mathsf{p}}|)$ is the fraction of ones in H. The event $A \wedge B$ happens iff tr_0 is accepting and it comes from a row $\bar{r} \in \mathsf{Rows}$ with at least $n + 1$ ones. Thus,

$$\Pr[A \wedge B] = \frac{\sum_{j=n+1}^{|\mathcal{F}_{\mathsf{p}}|} jR_j}{R \cdot |\mathcal{F}_{\mathsf{p}}|} = \frac{\sum_{j=1}^{|\mathcal{F}_{\mathsf{p}}|} jR_j}{R \cdot |\mathcal{F}_{\mathsf{p}}|} - \frac{\sum_{j=1}^{n} jR_j}{R \cdot |\mathcal{F}_{\mathsf{p}}|} = \Pr[A] - \frac{\sum_{j=1}^{n} jR_j}{R \cdot |\mathcal{F}_{\mathsf{p}}|} \ .$$

Next, $\sum_{j=1}^{n} jR_j$ is largest when all R rows have exactly n ones, i.e., $R_n = R$ and $R_j = 0$ for $j < n$. Thus, $(\sum_{j=1}^{n} jR_j)/(R \cdot |\mathcal{F}_{\mathsf{p}}|) \leq nR/(R \cdot |\mathcal{F}_{\mathsf{p}}|) = n/|\mathcal{F}_{\mathsf{p}}|$. Hence, $\Pr[A \wedge B] \geq \Pr[A] - n/|\mathcal{F}_{\mathsf{p}}|$. Finally, $\varepsilon_{\mathsf{ss}}(\mathsf{p}) \geq (\Pr[A] - n/|\mathcal{F}_{\mathsf{p}}|) + \Pr[\neg A] = 1 - n/|\mathcal{F}_{\mathsf{p}}|$.

Expected number of queries. Let Q denote the number of queries $\mathsf{Ext}_{\mathsf{rw}}$ makes to P^*. Since the total running time of $\mathsf{Ext}_{\mathsf{rw}}$ is $\mathsf{poly}(\lambda) \cdot \mathsf{Q}$, it is sufficient to only analyze Q. Consider the case that $H_{\bar{r},\alpha_0} = 1$ (therefore, A happened); then, the extractor in Fig. 2 will not abort on the second step but enters the **while** loop. The **while** loop in the extractor can be viewed as sampling without replacement from a finite binary-classified population.

Recall that the negative hypergeometric distribution (NHG) is the distribution of X in the next game: given a bin with N balls of which K are marked, X is the number of sampled balls from the bin (without replacements) until we get $k \leq K$ marked balls. The expected value of X is $\mathbb{E}[\mathrm{NHG}_{N,K,k}] = k(N + 1)/(K + 1)$. The number of iterations of the **while** loop corresponds to an NHG random variable with the following parameters, where $\delta_{\bar{r}}$ is the fraction of ones in $H_{\bar{r}}$ (i.e., row $\bar{r}$).

- $N = |\mathcal{F}_{\mathsf{p}}| - 1$ (the number of possible challenges except α_0),
- $K = \delta_{\bar{r}}|\mathcal{F}_{\mathsf{p}}| - 1$ (the number of ones in $H_{\bar{r}}$, except the entry $H_{\bar{r},\alpha_0}$),

– $k = n$ (the additional number of accepting transcripts $\mathsf{Ext}_{\mathsf{rw}}$ needs to find).

If there are at least $n + 1$ entries in the row $\bar{r}$, the expected number of iterations in the **while** loop to get n ones is $k(N+1)/(K+1) = n|\mathcal{F}_{\mathsf{p}}|/(\delta_{\bar{r}}|\mathcal{F}_{\mathsf{p}}|) = n/\delta_{\bar{r}}$. On the other hand, if there are less than $n+1$ ones in the row ϱ, then the **while** loop checks all the $|\mathcal{F}_{\mathsf{p}}|-1$ entries in the row. In this case, $K = \delta_{\bar{r}}|\mathcal{F}_{\mathsf{p}}|-1 \leq n-1$ and thus $|\mathcal{F}_{\mathsf{p}}| \leq n/\delta_{\bar{r}}$. Thus, the expected number of iterations in the **while** loop satisfies $|\mathcal{F}_{\mathsf{p}}| - 1 < n/\delta_{\bar{r}}$. Hence, $\mathbb{E}[\mathsf{Q} \mid A \wedge \bar{r}] \leq n/\delta_{\bar{r}}$.

The conditional probability, given the row $\bar{r}$, that the extractor does not abort before entering the **while** loop is $\delta_{\bar{r}} = \Pr[A \mid \bar{r}]$. Thus, conditioned on the row $\bar{r}$, the expected number of calls that the extractor makes is $\mathbb{E}[\mathsf{Q} \mid \bar{r}] \leq \delta_{\bar{r}} \cdot n/\delta_{\bar{r}} = n$. The expected number of calls of $\mathsf{Ext}_{\mathsf{rw}}$ can be computed as the average expected number of calls over the choice of $\bar{r}$:

$$\mathbb{E}[\mathsf{Q}] = \textstyle\sum_{\bar{r}\in\mathsf{Rows}} \mathbb{E}[\mathsf{Q}|\bar{r}] \Pr[\bar{r}] \leq \sum_{\varrho=1}^{|\mathsf{Rows}|} \frac{n}{|\mathsf{Rows}|} = n \enspace .$$

The claim follows. □

6 Black-Box Extractability

Definition. For a single transcript tr, we write that $(C, f) \in \mathcal{R}_{\mathsf{ck},\mathsf{tr}}$ iff $(C, f) \in \mathcal{R}_{\mathsf{ck}} \wedge f(\alpha) = \eta$. (This is a particular case of Eq. (1) for $n = 0$.)

Definition 6 (Black-Box Extractability). *A non-interactive polynomial commitment scheme* PC *is* black-box extractable *(BBE) for* Pgen*, if there exists an expected PPT black-box extractor* $\mathsf{Ext}_{\mathsf{bb}}$*, such that for all PPT* $\mathcal{A}$*, DPT* P^**,* $n \in \mathsf{poly}(\lambda)$*, and* $\mathcal{F}_{\mathsf{p}} \subseteq \mathbb{F}$ *of size* $|\mathcal{F}_{\mathsf{p}}| = \lambda^{\omega(1)}$*,* $\mathsf{Adv}^{\mathsf{bbe}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext},n,\mathcal{A},\mathsf{P}^*}(\lambda) =$

$$\Pr\left[\begin{array}{c} \mathsf{V}(\mathsf{ck},\mathsf{tr}_0) = 1 \wedge \\ (C, f) \notin \mathcal{R}_{\mathsf{ck},\mathsf{tr}_0} \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathsf{ck},\mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); (C, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{ck}); \\ \alpha_0 \leftarrow_{\$} \mathcal{F}_{\mathsf{p}}; (\eta_0, \pi_0) \leftarrow \mathsf{P}^*(\mathsf{st}, \alpha_0); \mathsf{tr}_0 \leftarrow (C, \alpha_0, \eta_0, \pi_0); \\ f \leftarrow \mathsf{Ext}_{\mathsf{bb}}^{\mathsf{P}^*(\mathsf{st},\cdot)}(\mathsf{ck},\mathsf{tr}_0); \end{array}\right] \approx_\lambda 0 \enspace .$$

$\mathsf{Ext}_{\mathsf{bb}}$ *can invoke* $\mathsf{P}^*(\mathsf{st}, \cdot)$ *with any challenge* $\tilde{\alpha} \in \mathcal{F}_{\mathsf{p}}$ *to which* $\mathsf{P}^*(\mathsf{st}, \tilde{\alpha})$ *returns some tuple* $(\tilde{\eta}, \tilde{\pi})$*.*

Note that st can contain information about ck and C. In Definition 6, P^* being deterministic means that we can rewind and restart P^* many times on the same state st, but different challenges.

We balanced Definition 6 so that it is weak enough for KZG to satisfy it and strong enough so that the SNARK compiler in Section 7 can use it. A crucial difference between Definition 1 and Definition 6 is that the evaluation point is sampled randomly from a large set in the latter. We need this property to prove Theorem 2. This difference with Definition 1 is why we must open each polynomial at a random evaluation point in our compiler in Section 7.

In Appendix C, we show that if the PCS is binding and BBE, it also has non-adaptive evaluation binding. This is a slightly weaker form of evaluation binding than usual, where the game picks the evaluation point randomly.

```
$\mathsf{Ext}_{\mathsf{bb}}^{\mathsf{P}^*(\mathsf{st},\cdot)}(\mathsf{ck}, \mathsf{tr}_0)$
1 :  if $\mathsf{V}(\mathsf{ck}, \mathsf{tr}_0) = 0$
       then return $\bot$; fi
2 :  $\mathbf{tr} \leftarrow \mathsf{Ext}_{\mathsf{rw}}^{\mathsf{P}^*(\mathsf{st},\cdot)}(\mathsf{ck}, \mathsf{tr}_0)$;
3 :  if $\exists j \in [1, n]. \mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 0$
       then return $\bot$; fi
     $f \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathsf{ck}, \mathsf{tr}_0 || \mathbf{tr})$;
4 :  return $f$;
```

```
$\mathcal{A}_{\mathsf{ss}}(\mathsf{ck})$
$(C, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{ck}); \alpha_0 \leftarrow\!\!\$\, \mathcal{F}_{\mathsf{p}}$;
$(\eta_0, \pi_0) \leftarrow \mathsf{P}^*(\mathsf{st}, \alpha_0)$;
$\mathsf{tr}_0 \leftarrow (C, \alpha_0, \eta_0, \pi_0)$;
if $\mathsf{V}(\mathsf{ck}, \mathsf{tr}_0) = 0$
  then return $\bot$; fi
$\mathbf{tr} \leftarrow \mathsf{Ext}_{\mathsf{rw}}^{\mathsf{P}^*(\mathsf{st},\cdot)}(\mathsf{ck}, \mathsf{tr}_0)$;
if $\exists j \in [1, n]. \mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 0$
  then return $\bot$; fi

return $(\mathsf{tr}_0 || \mathbf{tr})$;
```

Fig. 3. The black-box extractor $\mathsf{Ext}_{\mathsf{bb}}$ and the special-soundness adversary $\mathcal{A}_{\mathsf{ss}}^{\mathsf{P}^*(\mathsf{st},\cdot)}$ from Theorem 3.

Security Reduction. We prove that if a non-interactive polynomial commitment scheme is computationally special-sound, it is also black-box extractable, as in Definition 6. The proof of Theorem 3 per se is not complicated if one assumes the results of Sections 4 and 5.

Theorem 3. *If a non-interactive PCS* PC *is computational* $(n + 1)$*-special-sound, it is black-box extractable.*

Proof. Assume PC is special-sound (see Definition 5) and let $\mathsf{Ext}_{\mathsf{ss}}$ be the guaranteed PPT special soundness extractor. Let $\mathsf{Ext}_{\mathsf{rw}}$ be the rewinding extractor from Fig. 2. Let $\mathcal{A}$ and P^* be the adversaries in the definition of black-box extraction.

We depict the new black-box extractor $\mathsf{Ext}_{\mathsf{bb}}$ in Fig. 3. If the verifier does not accept tr_0, there is no need to extract, and $\mathsf{Ext}_{\mathsf{bb}}$ outputs $\bot$. Otherwise, $\mathsf{Ext}_{\mathsf{bb}}$ calls $\mathsf{Ext}_{\mathsf{rw}}$ to obtain n more accepting transcripts $\mathbf{tr}$, resulting in an $(n + 1)$-tuple $\mathsf{tr}_0 || \mathbf{tr}$ of admissible transcripts. Then, $\mathsf{Ext}_{\mathsf{bb}}$ runs $\mathsf{Ext}_{\mathsf{ss}}$ on $\mathsf{tr}_0 || \mathbf{tr}$, obtaining a polynomial f. If both $\mathsf{Ext}_{\mathsf{rw}}$ and $\mathsf{Ext}_{\mathsf{ss}}$ succeed, then $\mathsf{Ext}_{\mathsf{bb}}$ outputs f. Since $\mathsf{Ext}_{\mathsf{rw}}$ runs in the expected PPT and $\mathsf{Ext}_{\mathsf{ss}}$ in strict PPT, $\mathsf{Ext}_{\mathsf{bb}}$ runs in expected PPT.

We next bound the probability that when running $\mathsf{Ext}_{\mathsf{bb}}$, $\mathsf{V}(\mathsf{ck}, \mathsf{tr}_0) = 1$ but $(C, f) \notin \mathcal{R}_{\mathsf{ck},\mathsf{tr}_0}$. Looking at Fig. 3, we can see that there are two different ways for this to happen.

(i) $\mathsf{Ext}_{\mathsf{bb}}$ returns $\bot$ in step 3. This corresponds to the event $\mathbf{F}_{\mathsf{rw}}$ ("failure in rewinding") that $\mathsf{V}(\mathsf{ck}, \mathsf{tr}_0) = 1$ but $\mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 0$ for some $j \in [1, n]$.

(ii) $\mathsf{Ext}_{\mathsf{bb}}$ reaches step 4 and returns f, such that $(C, f) \notin \mathcal{R}_{\mathsf{ck},\mathsf{tr}_0}$. This corresponds to the event $\mathbf{F}_{\mathsf{ss}}$ ("failure in special soundness") that $\forall j \in [0, n]. \mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 1$ but $(C, f) \notin \mathcal{R}_{\mathsf{ck},\mathsf{tr}_0}$.

We next bound the probabilities that $\mathbf{F}_{\mathsf{rw}}$ or $\mathbf{F}_{\mathsf{ss}}$ happen.

Event $\mathbf{F}_{\mathsf{rw}}$. By Theorem 2, $\mathbf{F}_{\mathsf{rw}}$ happens with probability at most $n / |\mathcal{F}_{\mathsf{p}}|$.

Event $\mathbf{F}_{\mathsf{ss}}$. We bound the probability that $\mathbf{F}_{\mathsf{ss}}$ happens by relying on special soundness (Definition 5). In Fig. 3, we depict a special soundness adversary $\mathcal{A}_{\mathsf{ss}}$ that runs $\mathcal{A}$ and P^* internally. $\mathcal{A}_{\mathsf{ss}}$ first runs $\mathcal{A}$ and P^* to create tr_0 as in Definition 6. Crucially, $\mathcal{A}_{\mathsf{ss}}$ creates tr_0 from the correct distribution for $\mathsf{Ext}_{\mathsf{bb}}$. Then, $\mathcal{A}_{\mathsf{ss}}$ executes $\mathsf{Ext}_{\mathsf{bb}}$ up to step 3 Finally, $\mathcal{A}_{\mathsf{ss}}$ outputs tr_0 together with n transcripts computed by $\mathsf{Ext}_{\mathsf{rw}}$. (However, it does not run $\mathsf{Ext}_{\mathsf{ss}}$.)

According to the special soundness definition, the event that the extractor outputs $f(X)$ such that $(C, f) \notin \mathcal{R}_{\mathsf{ck},\mathsf{tr}_0}$, but $\forall j \in [0, n].\mathsf{V}(\mathsf{ck}, \mathsf{tr}_j) = 1$ is bounded $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n+1,\mathcal{A}_{\mathsf{ss}}}(\lambda)$. This corresponds precisely to the event $\mathbf{F}_{\mathsf{ss}}$. Thus, $\Pr[\mathbf{F}_{\mathsf{ss}}] \leq \mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n+1,\mathcal{A}_{\mathsf{ss}}}(\lambda)$. However, $\mathcal{A}_{\mathsf{ss}}$ is an expected PPT adversary, but special soundness is defined only for strict PPT adversary. We observe that special soundness is a falsifiable assumption (it can be written as an interaction between a PPT challenger and an adversary, Appendix A). According to Lemma 6, since $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n+1,\mathcal{A}'}(\lambda)$ is negligible for any PPT $\mathcal{A}'$, then also $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n+1,\mathcal{A}_{\mathsf{ss}}}(\lambda)$ is a negligible function.

Summing up the above results, we get the claim of the theorem. □

Corollary 1. *If* $(n+1)$-ARSDH *holds, then KZG for degree* $\leq n$ *polynomials is black-box extractable.*

Proof. Follows directly from Theorems 1 and 3. □

7 Application to SNARKs

7.1 Polynomial IOP

The following definition of polynomial IOP (PIOP) roughly follows [SZ20].

Definition 7 (Polynomial IOP with Preprocessing). *Let $\mathcal{R}$ be an indexed relation, $\mathbb{F}$ some finite field, and $d \in \mathbb{N}$ a degree bound. A* polynomial IOP *for $\mathcal{R}$ with degree bound d is a tuple of PPT algorithms* $\Pi_{\mathsf{IOP}} = (\mathsf{IOP.I}, \mathsf{IOP.P}, \mathsf{IOP.V})$, *satisfying the following description.*

- $\mathsf{IOP.I}$ *takes* $\mathbb{i}$ *for input and outputs a list of polynomials of degree at most d.*
- $\mathsf{IOP.V}$ *gets oracle access to these polynomials.*
- $(\mathsf{IOP.P}, \mathsf{IOP.V})$ *run a* Rnds*-round interactive protocol.*
- *In each round,* $\mathsf{IOP.P}$ *sends polynomials $f_i(X) \in \mathbb{F}[X]$ of degree at most d to* $\mathsf{IOP.V}$.
- $\mathsf{IOP.V}$ *is an oracle machine with access to a list of oracles, containing one oracle for each polynomial it has received from the prover (and indexer).*
- *At the end of each round,* $\mathsf{IOP.V}$ *can query oracles associated with a polynomial $f_i(X)$ on a point $\alpha_j \in \mathbb{F}$, the oracle responds with the value $f_i(\alpha_j)$.*
- *In the subsequent round,* $\mathsf{IOP.V}$ *sends oracle queries and challenges $z_k \in \mathbb{F}$ to* $\mathsf{IOP.P}$.
- $\mathsf{IOP.V}$ *is public coin.*

On common input $\mathbb{x}$ and prover's witness $\mathbb{w}$, the protocol transcript is denoted by the random variable $\langle \mathsf{IOP.P}(\mathbb{x}, \mathbb{w}), \mathsf{IOP.V}(\mathbb{x})\rangle = \mathsf{tr}$. When the verifier accepts the transcript, we conveniently write $\mathsf{tr} = 1$. $\mathsf{IOP.V}^{\mathrm{I}(\mathbb{i})}$ denotes that the verifier has oracle access to the polynomials outputted by the indexer.

A polynomial IOP must satisfy the following properties.
Perfect Completeness: for all $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$,

$$\Pr\left[\left\langle \mathsf{IOP.P}(\mathbb{x}, \mathbb{w}), \mathsf{IOP.V}^{\mathrm{I}(\mathbb{i})}(\mathbb{x})\right\rangle = 1\right] = 1 \ .$$

Knowledge Soundness: There exists a PPT algorithm Ext [8], such that for any interactive algorithm $\mathcal{A}$, $\mathbb{i} \in \mathbb{I}(\mathcal{R})$, and input $\mathbb{x} \in \{0,1\}^{\mathsf{poly}(\lambda)}$ (not necessarily in the language), the following holds: $\mathsf{Adv}^{\mathrm{ks}}_{\Pi,\mathsf{Ext},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\mathsf{tr} = 1 \wedge (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} \mid \mathsf{tr} \leftarrow \left\langle \mathcal{A}(\mathbb{i}, \mathbb{x}), \mathsf{IOP.V}^{\mathrm{I}(\mathbb{i})}(\mathbb{x})\right\rangle ; \mathbb{w} \leftarrow \mathsf{Ext}(\mathbb{i}, \mathbb{x}, \mathsf{tr})\right] \approx_\lambda 0 \ .$$

7.2 Compiling Polynomial IOPs Into Arguments

We present a compiler that compiles a PIOP and a non-interactive polynomial commitment scheme (as defined in Section 2.1) into an interactive argument.

The SRS is the public key of the commitment scheme, and in the preprocessing phase, the commitments of the indexer's polynomials are published. The interactive part of the argument runs PIOP, except with the following two changes:

- If the PIOP prover sends some polynomial f to the PIOP verifier, then P and V execute the following three-message subprotocol: the prover sends a commitment of f to the verifier, the verifier samples a new extraction point[9] $\chi \leftarrow_\$ \mathcal{F}$ (here $|\mathcal{F}| = \lambda^{\omega(1)}$), and the prover opens the commitment of f at this point. The verifier checks that the opening proof verifies. We can think of this protocol as an interactive commitment phase of a new polynomial commitment scheme KZG^+.
- If the PIOP verifier queries a previously sent polynomial f at some point α, the verifier sends α to the prover and the prover opens the commitment to f at the evaluation point α. The verifier checks that the opening proof verifies.

The rest of the PIOP is executed truthfully. In particular, the verifier checks that the PIOP verifier accepts the corresponding PIOP transcript that can be easily compiled from the verifier's responses and evaluations sent by the prover.

It is important to observe that even if the original PIOP has zero knowledge, the compiled argument might not have. We make new queries (with extraction points) to the polynomials, which leak additional information. However, in efficient zk-SNARKs like Plonk, this can be tackled by adding extra randomness to polynomials to account for one more opening point.

[8] Note that [BFS20] allowed the extractor to rewind. We are unaware of any concrete polynomial IOP, which would need that. Thus, for the sake of simplicity we use a straight-line extractor.

[9] In practice, $\mathcal{F} = \mathbb{F}$. After using Fiat-Shamir, the extraction point is a hash.

For simplicity, we assume that the PIOP prover sends a single polynomial in each round. This assumption is without loss of generality: We can always write an equivalent PIOP, where instead of sending many polynomials per round, the prover sends each in a separate round, and the verifier's intermediate responses are random challenges never used by the prover. In the compiled argument, these challenges can be removed, meaning there is no increase in the number of rounds.

Compiler Description. We introduce more notation to make it easier to explain rewinding later on. Let $(\mathsf{IOP.I}, \mathsf{IOP.P}, \mathsf{IOP.V})$ be a polynomial IOP. We denote by $\mathsf{st}^{\mathsf{V}}_{r-1}$ the verifier's state at the beginning of round r. The verifier's initial state is $\mathsf{st}^{\mathsf{V}}_0 = \mathbb{x}$. On input $(\mathsf{st}^{\mathsf{V}}_{r-1}, \mathsf{coins}_r)$, V outputs query points $\boldsymbol{\alpha}_r$, other challenge values $\boldsymbol{z}_r$, and labels[10] $\boldsymbol{g}_r$ for the oracles that need to be evaluated. More precisely, $\boldsymbol{g}_r$ is a vector of length $\ell_r = |\boldsymbol{g}_r| = |\boldsymbol{\alpha}_r|$ containing labels for V's oracles that need to be queried. An oracle can be from the prover or from the indexer (the same oracle may appear even multiple times in $\boldsymbol{g}_r$). The verifier queries $\eta_{r,i} \leftarrow g_{r,i}(\alpha_{r,i})$ for all $i \in [1, \ell_r]$, where $g_{r,i}$ is a label for one of V's oracles.

We depict the compiler, with full details, in Fig. 4. Here, $\Pi_{\mathsf{IOP}} = (\mathsf{IOP.I}, \mathsf{IOP.P}, \mathsf{IOP.V})$ is a Rnds-round PIOP for $\mathcal{R}_{\mathsf{p},n}$ and PC is a non-interactive polynomial commitment scheme.

Security. We prove that if the PIOP is knowledge sound, the commitment scheme is evaluation binding and black-box extractable, then the compiled (interactive) argument system has witness-extended emulation.

Theorem 4. *If the non-interactive polynomial commitment scheme* PC *is black-box extractable and evaluation-binding, and if the* Rnds*-round PIOP* Π_{IOP} *for* $\mathcal{R}$ *is knowledge-sound, then the argument* Π *described in Fig. 4 is a public-coin interactive argument system for* $\mathcal{R}$ *that has witness-extended emulation.*

Proof. Let P^*, $\mathcal{A}$, $\mathcal{D}$ denote the adversaries from the WEE definition (Section 2.2). We construct a WEE emulator $\Pi.\mathsf{Emu}$ for the argument system Π. $\Pi.\mathsf{Emu}$ is defined in Fig. 5, having black-box rewindable access to P^*.

We denote the state of the prover P^* (containing the state st from $\mathcal{A}$, P^*'s internal configuration, and messages from V) at the beginning of the round r subprotocol by $\mathsf{st}^{(1)}_{r-1}$ and after receiving the challenge $\chi_r \leftarrow_{\$} \mathcal{F}$ by $\mathsf{st}^{(2)}_{r-1}$. Note that $\Pi.\mathsf{Emu}$ has only an oracle access to P^* and does not see P^*'s state. We use the state notation only for showing that $\Pi.\mathsf{Emu}$ rewinds the deterministic P^* to a particular state.

We give an overview of $\Pi.\mathsf{Emu}$ depicted in Fig. 5. $\Pi.\mathsf{Emu}$ black-box executes the prover's first round $\mathsf{P}^*(\mathsf{st}^{(1)}_0)$ and gets a commitment C_1 to an unknown polynomial. The initial state is $\mathsf{st}^{(1)}_0 = (\mathsf{p}, \mathsf{ck}, \mathbb{x}, \mathsf{st})$, where st is the state information sent from $\mathcal{A}$. Then, $\Pi.\mathsf{Emu}$ plays the argument's verifier, sampling $\chi_1 \leftarrow_{\$} \mathcal{F}$ and

[10] They have to be labels, not polynomials, since technically V does not know the polynomials themselves. For simplicity, we will ignore this difference in the notation.

$\mathsf{Pgen}(1^\lambda)$: run $\mathsf{p} \leftarrow \mathsf{PC.Pgen}(1^\lambda)$ of the PCS.

$\mathsf{KGen}(\mathsf{p}, n)$: run $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{PC.KGen}(\mathsf{p}, n)$.

$\mathsf{Derive}(\mathsf{ck}, \mathbb{i} \in \mathbb{I}(\mathcal{R}_{\mathsf{p},n}))$: run $(\iota_1(X), \dots, \iota_\ell(X)) \leftarrow \mathsf{IOP.I}(\mathbb{i})$ and compute $c_j \leftarrow \mathsf{PC.Com}(\mathsf{ck}, \iota_j(X))$ for $j = 1, \dots, \ell$. Return $\mathsf{ek}_{\mathbb{i}} \leftarrow \mathsf{ck}$ and $\mathsf{vk}_{\mathbb{i}} \leftarrow (c_1, \dots, c_\ell)$.

$\langle \mathsf{P}(\mathsf{ek}_{\mathbb{i}}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}) \rangle$:

- For each $r \in [1, \mathsf{Rnds}]$, P and V run the following subprotocol:
  1. P sends $C_r \leftarrow \mathsf{PC.Com}(\mathsf{ck}, f_r)$.
     // where $f_r \in \mathbb{F}_{\leq n}[X]$ is sent by $\mathsf{IOP.P}$ in the r-th PIOP round
  2. V sends $\chi_r \leftarrow_\$ \mathcal{F}$.
  3. P sends $(\eta_{\chi_r}, \pi_{\chi_r}) \leftarrow \mathsf{PC.Open}(\mathsf{ck}, C_r, \chi_r, f_r)$.
  4. V sends random coins coins_r of $\mathsf{IOP.V}$.
  5. P computes $(\boldsymbol{z}_r, \boldsymbol{\alpha}_r, \boldsymbol{g}_r) \leftarrow \mathsf{IOP.V}(\mathsf{st}^{\mathsf{V}}_{r-1}, \mathsf{coins}_r)$ and sends a vector of openings $(\boldsymbol{\eta}_r, \boldsymbol{\pi}_r)$, where $(\eta_{r,i}, \pi_{r,i}) \leftarrow \mathsf{PC.Open}(\mathsf{ck}, C_r, \alpha_{r,i}, g_{r,i})$ for each $i \in [1, \ell_r]$. // $\ell_r = |\boldsymbol{g}_r| = |\boldsymbol{\alpha}_r|$.
  6. P and V update their states.
- V uses $\mathsf{PC.V}$ to check that all the openings are valid and $\mathsf{IOP.V}$ to check that the opened PIOP transcript is valid.

Fig. 4. Public-coin interactive argument system $\Pi = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Derive}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for the relation $\mathcal{R}_{\mathsf{p},n}$.

running $\mathsf{P}^*(\mathsf{st}_0^{(1)}, \chi_1)$ to obtain an opening $(\eta_{\chi_1}, \pi_{\chi_1})$. If $\Pi.\mathsf{Emu}$ received a valid opening for χ, it runs the black-box extractor $\mathsf{PC.Ext_{bb}}$ of PC to obtain a polynomial f_1, such that $(C_1, f_1) \in \mathcal{R}_{\mathsf{ck}}$. We show later that Ext extracts a valid polynomial with overwhelming probability. This follows from the definition of black-box extractability (Definition 6), which can be invoked because χ_1 is an extraction point sampled from a superpolynomial size set. Then, $\Pi.\mathsf{Emu}$ samples coins_1 for $\mathsf{IOP.V}$ and uses them to compute evaluation points $\boldsymbol{\alpha}_1$, challenges $\boldsymbol{z}_1$, and oracle labels $\boldsymbol{g}_1$ that have to be evaluated on $\boldsymbol{\alpha}_1$. It runs P^* on coins_1 to get openings $(\boldsymbol{\eta}_1, \boldsymbol{\pi}_1)$.

If f_1 was correctly extracted, $\Pi.\mathsf{Emu}$ computes $\eta_{1,i} \leftarrow g_{1,i}(\alpha_{1,i})$, evaluating all the polynomials that the verifier indicated ($\boldsymbol{g}_1$ may include a label for f_1 or indexer's polynomials) at the respective evaluation points. $\Pi.\mathsf{Emu}$ checks if the openings returned by P^* are compatible with the one it can compute itself using f_1. If so, then $\Pi.\mathsf{Emu}$ has successfully simulated the first round of the PIOP, $(f_1, \boldsymbol{\alpha}_1, \boldsymbol{z}_1, \boldsymbol{g}_1)$. Otherwise, if one of the openings is incompatible, we get a contradiction with the evaluation binding property of PC, as we prove later. After updating the state of $\mathsf{IOP.V}$, $\Pi.\mathsf{Emu}$ proceeds to simulate the subsequent rounds similarly. Finally, the emulator runs the PIOP extractor on the PIOP transcript to recover the witness $\mathbb{w}$. $\Pi.\mathsf{Emu}$ outputs the argument's transcript and $\mathbb{w}$.

```
$\mathit{\Pi}.\mathsf{Emu}^{\langle \mathsf{P}^*(\mathsf{p},\mathsf{ck},\mathbb{x},\mathsf{st}),\mathsf{V}(\mathsf{vk}_{\mathbb{i}},\mathbb{x})\rangle}(\mathsf{p},\mathsf{ck},\mathbb{x})$
$\mathsf{colsn} \leftarrow$ **false**;
**for** $r = 1$ to $\mathsf{Rnds}$ **do**   // $\mathsf{P}^*$'s initial state is $\mathsf{st}_0^{(1)} = (\mathsf{p},\mathsf{ck},\mathbb{x},\mathsf{st})$
    Run $\mathsf{P}^*(\mathsf{st}_{r-1}^{(1)})$ and receive $C_r$;
    Sample extraction point $\chi_r \leftarrow_\$ \mathcal{F}$;
    $(\eta_{\chi_r}, \pi_{\chi_r}) \leftarrow \mathsf{P}^*(\mathsf{st}_{r-1}^{(1)}, \chi_r)$;   // $P^*$'s new state is denoted $\mathsf{st}_{r-1}^{(2)}$
    $\mathsf{tr}_{\mathsf{PC}} \leftarrow (C_r, \chi_r, \eta_{\chi_r}, \pi_{\chi_r})$;
    **if** $\mathsf{PC.V}(\mathsf{ck},\mathsf{tr}_{\mathsf{PC}}) = 1$ **then**
        $f_r(X) \leftarrow \mathsf{PC.Ext}_{\mathsf{bb}}^{\mathsf{P}^*(\mathsf{st}_{r-1}^{(1)},\cdot)}(\mathsf{ck},\mathsf{tr}_{\mathsf{PC}})$;
        **if** $(C_r, f_r) \in \mathcal{R}_{\mathsf{ck}}$ **then**
            $\mathsf{ext\text{-}fail}_r \leftarrow$ **false**;
        **else** $\mathsf{ext\text{-}fail}_r \leftarrow$ **true**; **fi**
    **fi**
    Sample $\mathsf{coins}_r$ for $\mathsf{IOP.V}$; $(\boldsymbol{\alpha}_r, \boldsymbol{z}_r, \boldsymbol{g}_r) \leftarrow \mathsf{IOP.V}(\mathsf{st}_{r-1}^{\mathsf{V}}, \mathsf{coins}_r)$;
    $(\boldsymbol{\eta}_r, \boldsymbol{\pi}_r) \leftarrow \mathsf{P}^*(\mathsf{st}_{r-1}^{(2)}, \mathsf{coins}_r)$;   // $P^*$'s new state is denoted $\mathsf{st}_r^{(1)}$
    $\mathsf{st}_r^{\mathsf{V}} \leftarrow (\mathsf{st}_{r-1}^{\mathsf{V}}, \mathsf{coins}_r, \boldsymbol{\eta}_r, \boldsymbol{\pi}_r)$;
    **if** $(\mathsf{PC.V}(\mathsf{ck},\mathsf{tr}_{\mathsf{PC}}) = 1) \wedge \neg(\mathsf{ext\text{-}fail}_r)$ **then**
        Compute $\tilde{\boldsymbol{\eta}}_r \leftarrow (g_{r,1}(\alpha_{r,1}), \ldots, g_{r,\ell_r}(\alpha_{r,\ell_r}))$;
        **if** $\boldsymbol{\eta}_r \neq \tilde{\boldsymbol{\eta}}_r$ **then** $\mathsf{colsn} \leftarrow$ **true**; **fi**
    **fi**
**endfor**
$\mathsf{tr} \leftarrow (\mathsf{tr}_1, \ldots, \mathsf{tr}_{\mathsf{Rnds}})$, where $\mathsf{tr}_i = (C_i, \chi_i, \eta_{\chi_i}, \pi_{\chi_i}, \mathsf{coins}_i, \boldsymbol{\eta}_i, \boldsymbol{\pi}_i)$;
$\mathsf{tr}_{\mathsf{IOP}} \leftarrow (f_1, \boldsymbol{\alpha}_1, \boldsymbol{z}_1, \boldsymbol{g}_1, \ldots, f_{\mathsf{Rnds}}, \boldsymbol{\alpha}_{\mathsf{Rnds}}, \boldsymbol{z}_{\mathsf{Rnds}}, \boldsymbol{g}_{\mathsf{Rnds}})$;
**if** $\vee_{i=1}^{\mathsf{Rnds}} \mathsf{ext\text{-}fail}_i$ **then return** $(\mathsf{tr}, \perp)$; **fi**
**if** $\mathsf{colsn}$ **then return** $(\mathsf{tr}, \perp)$; **fi**
$\mathbb{w} \leftarrow \mathsf{Ext}(\mathbb{x}, \mathsf{tr}_{\mathsf{IOP}})$, where $\mathsf{Ext}$ is the PIOP knowledge extractor.
**if** $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$ **then return** $(\mathsf{tr}, \perp)$; **fi**
**return** $(\mathsf{tr}, \mathbb{w})$;
```

Fig. 5. Emulator $\mathit{\Pi}.\mathsf{Emu}$ for the compiled argument $\mathit{\Pi}$.

Note that the transcript that $\mathit{\Pi}.\mathsf{Emu}$ outputs is distributed as the one resulted from an honest protocol execution between P^* and V. Therefore, real and emulated transcripts are perfectly indistinguishable. Let $\mathbf{D}_0$ be the event "$\mathbb{i} \in \mathbb{I}(\mathcal{R}_{\mathsf{p},n}) \wedge \mathcal{D}(\mathsf{tr}) = 1$" with the real transcript tr and $\mathbf{D}_1$ the same event when tr is emulated. Additionally, let $\mathbf{V}$ be the event that "$\mathsf{V}_{\mathsf{check}}(\mathsf{ck},\mathsf{tr}) = 1$" and $\mathbf{W}$ the event $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ when the emulator outputs tr and $\mathbb{w}$. Then,

$$\begin{aligned}\mathsf{Adv}^{\mathrm{wee}}_{\mathsf{Pgen},\mathit{\Pi},n,\mathsf{P}^*,\mathcal{A},\mathcal{D}}(\lambda) &= \Pr[\mathbf{D}_0] - \Pr[\mathbf{D}_1 \wedge (\mathbf{V} \Rightarrow \mathbf{W})] \\ &= \Pr[\mathbf{D}_1] - \Pr[\mathbf{D}_1 \wedge (\neg\mathbf{V} \vee \mathbf{W})] \\ &= \Pr[\mathbf{D}_1 \wedge \mathbf{V} \wedge \neg\mathbf{W}] \leq \Pr[\mathbf{V} \wedge \neg\mathbf{W}] \enspace .\end{aligned}$$

Thus, to conclude the proof, we need to bound the probability that $\Pi.\mathsf{Emu}$ computes a valid transcript but fails to extract a witness. This can be divided into the following disjoint events.

1. $\Pi.\mathsf{Emu}$ computes a valid transcript, but on the line 23 in Fig. 5 it returns $(\mathsf{tr}, \perp)$. Namely, there exists r, for which $\mathsf{PC.Ext_{bb}}$ failed to extract a polynomial f_r such that $(C_r, f_r) \in \mathcal{R}_{\mathsf{ck}}$, and consequently the corresponding variable $\mathsf{ext\text{-}fail}_r$ is set to **true**. We denote this event $\mathbf{F}^{(r)}_{\mathsf{bbe}}$ (failure in the black box extraction). Let $\mathbf{F}_{\mathsf{bbe}} = \mathbf{F}^{(1)}_{\mathsf{bbe}} \vee \ldots \vee \mathbf{F}^{(\mathsf{Rnds})}_{\mathsf{bbe}}$ be the event that extraction fails in any of the rounds.
2. $\Pi.\mathsf{Emu}$ computes a valid transcript, but on the line 24 in Fig. 5 it returns $(\mathsf{tr}, \perp)$. We denote the event by $\mathbf{F}_{\mathsf{eb}}$ (failure in evaluation binding). In that case $\mathsf{ext\text{-}fail}_r = \mathsf{false}$ for all r, but $\mathsf{colsn} = \mathsf{true}$. The variable colsn is set to true if some evaluation $\eta_{r,\alpha}$ sent by P^* is different from $\tilde{\eta}_{r,\alpha} = g_{r,i}(\alpha_{r,i})$, where $g_{r,i}(X)$ is either a publicly-known indexer polynomial or one of the extracted polynomials and $\alpha_{r,i} \in \boldsymbol{\alpha}_r$ is one of the query points.
3. $\Pi.\mathsf{Emu}$ computes a valid transcript, but on the line 26 in Fig. 5 it returns $(\mathsf{tr}, \perp)$. We denote this event by $\mathbf{F}_{\mathsf{iop}}$ (failure in the PIOP extraction). In that case, $\mathsf{tr_{IOP}}$ compiled by the emulator is a valid PIOP transcript because polynomials have been successfully extracted, and there are no collisions, but the PIOP extractor still failed to extract the witness.

We prove that each of the three events happens with a negligible probability. We bound the first event by the probability of breaking black-box extractability. We show this first for a fixed round r.

Lemma 3. *There exists a expected PPT $\mathcal{A}_r$ such that* $\Pr\left[\mathbf{F}^{(r)}_{\mathsf{bbe}}\right] \leq \mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen,PC,Ext},n,\mathcal{A}_r,\mathsf{P}^*}(\lambda)$.

Proof. We construct a reduction to the black-box extractability property of PC. Let $\mathcal{A}$, P^* be the adversaries in the WEE definition as above.

To construct a reduction, we need to construct a black-box extractability adversary $\mathcal{A}_r$ (we use P^* as the deterministic black-box extractability adversary). The adversary $\mathcal{A}_r$ runs $\Pi.\mathsf{Emu}$ until the line 2 in r-th execution of **for** loop. That is, until the emulator gets C_r. At that point, $\mathcal{A}_r$ outputs $(C_r, \mathsf{st}^{(1)}_{r-1})$.

We now plug $\mathcal{A}_r$ and P^* into the black-box extractability game. Let us recall the game. The adversary $\mathcal{A}_r(\mathsf{ck})$ outputs $(C_r, \mathsf{st}^{(1)}_{r-1})$ on a correctly sampled ck. Then the game picks $\chi \leftarrow_{\$} \mathcal{F}_{\mathsf{p}}$, sets $(\eta_{r,\chi}, \pi_{r,\chi}) \leftarrow \mathsf{P}^*(\mathsf{st}^{(1)}_{r-1}, \chi)$ and defines $\mathsf{tr_{PC}} \leftarrow (C_r, \chi, \eta_{r,\chi}, \pi_{r,\chi})$. Finally, $\mathsf{PC.Ext}_{\mathsf{bb}}^{\mathsf{P}^*(\mathsf{st}^{(1)}_{r-1},\cdot)}(\mathsf{ck}, \mathsf{tr_{PC}})$ outputs a polynomial $f_r(X)$. According to the definition, probability that $\mathsf{PC.V}(\mathsf{ck}, \mathsf{tr_{PC}}) = 1$ and $(C_r, f_r) \notin \mathcal{R}_{\mathsf{ck,tr}}$ is bounded by $\mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen,PC,Ext},n,\mathcal{A}_r,\mathsf{P}^*}(\lambda)$.

By construction of $\mathcal{A}_r$ and P^* the latter event is implied by the event $\mathbf{F}_{\mathsf{bbe}}$. Therefore, $\Pr[\mathbf{F}_{\mathsf{bbe}}] \leq \mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen,PC,Ext},n,\mathcal{A}_r,\mathsf{P}^*}(\lambda)$. $\square$

From the above it follows that $\Pr[\mathbf{F}_{\mathsf{bbe}}] \leq \sum_{r=1}^{\mathsf{Rnds}} \mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen,PC,Ext},n,\mathcal{A}_r,\mathsf{P}^*}(\lambda) \leq \mathsf{Rnds} \cdot \mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen,PC,Ext},n,\bar{\mathcal{A}},\mathsf{P}^*}(\lambda)$ for some $\bar{\mathcal{A}} \in \{\mathcal{A}_1, \ldots, \mathcal{A}_{\mathsf{Rnds}}\}$.

The second type of abortion can only happen with negligible probability due to the evaluation binding property (Section 2.1) of PC, which requires that it is computationally hard to open a commitment for two different evaluations at the same point. We prove it in the following lemma.

Lemma 4. *There exists a expected PPT $\mathcal{B}$ such that* $\Pr[\mathbf{F}_{\mathsf{eb}}] \leq \mathsf{Adv}^{\mathrm{evbind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{B}}(\lambda)$.

Proof. Recall that if $\Pi.\mathsf{Emu}$ aborts on the line 24, then $\mathsf{colsn} = \mathsf{true}$, but $\mathsf{ext\text{-}fail}_r = \mathsf{false}$ for all $r \in [1, \mathsf{Rnds}]$. It means that in each subprotocol the extraction of the polynomial was successful (for all r, $(C_r, f_r) \in \mathcal{R}_{\mathsf{ck}}$), but for some rth subprotocol $\eta_{r,\alpha} \neq \tilde{\eta}_{r,\alpha} = g_{r,i}(\alpha_{r,i})$ for some $i \in [1, \ell_r]$ and a polynomial $g_{r,i}$, which is either one of the extracted polynomials or a publicly known indexer polynomial.

Given $\Pi.\mathsf{Emu}$ as defined in Fig. 5, we construct an adversary $\mathcal{B}$ that breaks evaluation binding whenever the event $\mathbf{F}_{\mathsf{eb}}$ happens. $\mathcal{B}$ gets $(\mathsf{p}, \mathsf{ck})$ as input and starts by internally running the WEE adversary $\mathcal{A}(\mathsf{p}, \mathsf{ck})$ to obtain $(\mathbb{x}, \mathsf{st})$. After this $\mathcal{B}$ runs $\Pi.\mathsf{Emu}$ until at some round r, P^* outputs $(\boldsymbol{\pi}_r, \boldsymbol{\eta}_r)$ such that $\boldsymbol{\eta}_r \neq \tilde{\boldsymbol{\eta}}_r$. In this case, there exists $\alpha_{r,i} \in \boldsymbol{\alpha}_r$ such that $\eta_{r,i} \neq \tilde{\eta}_{r,i} = g_{r,i}(\alpha_{r,i})$ for some $(C, g_{r,i}) \in \mathcal{R}_{\mathsf{ck}}$. Here, C could be a commitment produced by an indexer (in that case $g_{r,i}(X)$ is publicly known) or one of the commitments sent by the prover (in that case $(C, g_{r,i}) \in \mathcal{R}_{\mathsf{ck}}$ since the extraction succeeded). $\mathcal{B}$ computes an opening proof $\tilde{\pi}_{r,i}$ using $\mathsf{PC.Open}(\mathsf{ck}, C, \alpha_{r,i}, g_{r,i})$ and outputs $(C, \alpha_{r,i}, \eta_{r,i}, \pi_{r,i}, \tilde{\eta}_{r,i}, \tilde{\pi}_{r,i})$.

The event $\mathbf{F}_{\mathsf{eb}}$ implies the following:

- The argument's verifier V accepts, thus $\mathsf{PC.V}(\mathsf{ck}, C, \alpha_{r,i}, \eta_{r,i}, \pi_{r,i}) = 1$.
- The extraction of polynomials was successful and thus $(C, g_{r,i}) \in \mathcal{R}_{\mathsf{ck}}$. By the completeness property of the commitment scheme, $\mathsf{PC.V}(\mathsf{ck}, C, \alpha_{r,i}, \tilde{\eta}_{r,i}, \tilde{\pi}_{r,i}) = 1$.
- Since $\mathsf{colsn} = \mathbf{true}$, $\eta_{r,i} \neq \tilde{\eta}_{r,i}$.

It follows that $\mathcal{B}$ breaks evaluation binding when the event $\mathbf{F}_{\mathsf{eb}}$ happens. We get $\Pr[\mathbf{F}_{\mathsf{eb}}] \leq \mathsf{Adv}^{\mathrm{evbind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{B}}(\lambda)$. □

Recall, $\mathbf{F}_{\mathsf{iop}}$ is the event that $\mathsf{tr}_{\mathsf{IOP}}$ is accepting, but the PIOP extractor in Step 25 returns $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$. We bound the probability of this event in the following lemma.

Lemma 5. *There exists an interactive algorithm $\mathcal{C}$ (potentially unbounded) such that* $\Pr[\mathbf{F}_{\mathsf{iop}}] \leq \mathsf{Adv}^{\mathrm{ks}}_{\Pi_{\mathsf{IOP}},\mathsf{Ext},\mathcal{C}}(\lambda)$.

Proof. The knowledge soundness of PIOP holds for any fixed $\mathbb{i} \in \mathbb{I}(\mathcal{R})$ and $\mathbb{x} \in \{0,1\}^{\mathsf{poly}(\lambda)}$. If it holds for any fixed input, then it will also hold when $\mathbb{x}$ and $\mathbb{i}$ are sampled. We sample $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n)$, and $(\mathbb{i}, \mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{ck})$.

We will construct an adversary $\mathcal{C}$ against the knowledge soundness of PIOP that takes $(\mathbb{i}, \mathbb{x})$ as an input. The adversary $\mathcal{C}$ runs identically to the **for** loop in $\Pi.\mathsf{Emu}$ except for two notable difference. First, instead of sampling coins_r by

```
$\langle \mathcal{C}(\mathbb{i}, \mathbb{x}), \mathsf{V}^{\mathrm{I}(\mathbb{i})}(\mathbb{x}) \rangle$   // $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$; $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n)$; $(\mathbb{i}, \mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{ck})$
1 :  for $r = 1$ to $\mathsf{Rnds}$ do   // $\mathsf{P}^*$'s initial state is $\mathsf{st}_0^{(1)} = (\mathsf{p}, \mathsf{ck}, \mathbb{x}, \mathsf{st})$
2 :    Run $\mathsf{P}^*(\mathsf{st}_{r-1}^{(1)})$ and receive $C_r$;
3 :    Sample extraction point $\chi_r \leftarrow\$ \mathcal{F}$;
4 :    $(\eta_{\chi_r}, \pi_{\chi_r}) \leftarrow \mathsf{P}^*(\mathsf{st}_{r-1}^{(1)}, \chi_r)$;   // $P^*$'s new state is denoted $\mathsf{st}_{r-1}^{(2)}$
5 :    $\mathsf{tr}_{\mathsf{PC}} \leftarrow (C_r, \chi_r, \eta_{\chi_r}, \pi_{\chi_r})$;
6 :    if $\mathsf{PC.V}(\mathsf{ck}, \mathsf{tr}_{\mathsf{PC}}) = 1$ then
7 :      $f_r(X) \leftarrow \mathsf{PC.Ext}_{\mathsf{bb}}^{\mathsf{P}^*(\mathsf{st}_{r-1}^{(1)}, \cdot)}(\mathsf{ck}, \mathsf{tr}_{\mathsf{PC}})$;
8 :      if $(C_r, f_r) \in \mathcal{R}_{\mathsf{ck}}$ then
9 :        Send $f_r(X)$ to $\mathsf{IOP.V}$; Receive $(\boldsymbol{\alpha}_r, \boldsymbol{z}_r, \boldsymbol{g}_r)$ from $\mathsf{IOP.V}$;
10 :        Compute $\mathsf{coins}_r$ such that $(\boldsymbol{\alpha}_r, \boldsymbol{z}_r, \boldsymbol{g}_r) = \mathsf{IOP.V}(\mathsf{st}_{r-1}^{\mathsf{V}}, \mathsf{coins}_r)$;
11 :        $(\boldsymbol{\eta}_r, \boldsymbol{\pi}_r) \leftarrow \mathsf{P}^*(\mathsf{st}_{r-1}^{(2)}, \mathsf{coins}_r)$;   // $P^*$'s new state is denoted $\mathsf{st}_r^{(1)}$
12 :        Compute $\tilde{\boldsymbol{\eta}}_r \leftarrow (g_{r,1}(\alpha_{r,1}), \ldots, g_{r,\ell_r}(\alpha_{r,\ell_r}))$;
13 :        if $\boldsymbol{\eta}_r \neq \tilde{\boldsymbol{\eta}}_r$ then abort; fi
14 :      else abort; fi
15 :    else abort; fi
16 :    $\mathsf{st}_r^{\mathsf{V}} \leftarrow (\mathsf{st}_{r-1}^{\mathsf{V}}, \mathsf{coins}_r, \boldsymbol{\eta}_r, \boldsymbol{\pi}_r)$;
17 :  endfor
```

Fig. 6. The adversary $\mathcal{C}$ against knowledge soundness of PIOP. We have *highlighted* the main differences compared to $\Pi.\mathsf{Emu}$.

itself, it sends f_r to $\mathsf{IOP.V}$ and receives back $(\boldsymbol{\alpha}_r, \boldsymbol{z}_r, \boldsymbol{g}_r)$. It then computes coins_r corresponding to $(\boldsymbol{\alpha}_r, \boldsymbol{z}_r, \boldsymbol{g}_r)$. We assume that this is efficient since we deal with public coin protocols. Second, it aborts the protocol if either the verification of the commitment opening fails, extraction of a polynomial fails ($\mathsf{ext\text{-}fail}_r = \mathbf{true}$), or there is a collision ($\mathsf{colsn} = \mathbf{true}$). Complete details of $\mathcal{C}$ can be found in Fig. 6.

In case the extraction of f_r is successful in each round and there are no collisions, the resulting PIOP transcript $\mathsf{tr}_{\mathsf{IOP}}$ is accepting. The probability of the PIOP extractor failing in this case corresponds precisely to the event $\mathbf{F}_{\mathsf{iop}}$. Thus, $\Pr[\mathbf{F}_{\mathsf{iop}}] \leq \mathsf{Adv}^{\mathrm{ks}}_{\Pi_{\mathsf{IOP}}, \mathsf{Ext}, \mathcal{C}}(\lambda)$. □

By combining the results of all the lemmas above, we get that $\Pr[\mathbf{V} \wedge \neg \mathbf{W}] \leq \mathsf{Rnds} \cdot \mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen}, \mathsf{PC}, \mathsf{Ext}, n, \bar{\mathcal{A}}, \mathsf{P}^*}(\lambda) + \mathsf{Adv}^{\mathrm{evbind}}_{\mathsf{Pgen}, \mathsf{PC}, n, \mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{ks}}_{\Pi_{\mathsf{IOP}}, \mathsf{Ext}, \mathcal{C}}(\lambda)$. Finally, we must consider that $\bar{\mathcal{A}}$ and $\mathcal{B}$ are expected PPT algorithms, but BBE and evaluation binding holds against strict PPT adversaries.

Both BBE and evaluation binding can be viewed as falsifiable assumptions. They can be written as an interaction between an efficient challenger and an adversary (see Appendix A for a formal definition). According to assumptions of our theorem, for any (strict) PPT $\mathcal{A}'$ and $\mathcal{B}'$, $\mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen}, \mathsf{PC}, \mathsf{Ext}, n, \mathcal{A}', \mathsf{P}^*}(\lambda)$ and $\mathsf{Adv}^{\mathrm{evbind}}_{\mathsf{Pgen}, \mathsf{PC}, n, \mathcal{B}'}(\lambda)$ are negligible functions in λ. Therefore, by applying Lemma 6, $\mathsf{Adv}^{\mathrm{bbe}}_{\mathsf{Pgen}, \mathsf{PC}, \mathsf{Ext}, n, \bar{\mathcal{A}}, \mathsf{P}^*}(\lambda)$ and $\mathsf{Adv}^{\mathrm{evbind}}_{\mathsf{Pgen}, \mathsf{PC}, n, \mathcal{B}}(\lambda)$ are also negligible. The claim of the theorem follows since $\mathsf{Adv}^{\mathrm{ks}}_{\Pi_{\mathsf{IOP}},\mathsf{Ext},\mathcal{C}}(\lambda)$ is negligible according to the knowledge soundness of PIOP. □

As previously argued, KZG PCS has black-box extractability and evaluation binding under the ARSDH assumption. Thus, we get the following result.

Corollary 2. *If the* ARSDH *assumption holds and if a* Rnds*-round Polynomial IOP for* $\mathcal{R}$ *has negligible knowledge error, then the argument* Π *described in Fig. 4, instantiated with KZG PCS, is a public-coin interactive argument for* $\mathcal{R}$ *that has witness-extended emulation.*

Finally, if we use, for example, Plonk's PIOP, then we obtain a public coin interactive argument with a constant proof size under a falsifiable assumption. The resulting argument retains Plonk's original asymptotic efficiency features. Below, $O_\lambda(\cdot)$ is the common "Big-O" notation, but we ignore $\mathsf{poly}(\lambda)$ factors.

Corollary 3. *Let* N *be the number of gates in an arithmetic circuit that describes a relation* $\mathcal{R}$*, and let* $|x|$ *denote the statement size. If the* ARSDH *assumption holds, then there exists a public coin argument system for* $\mathcal{R}$ *with* $O_\lambda(1)$ *proof size,* $O_\lambda(N)$ *SRS size,* $O_\lambda(|x|+\log N)$ *verifier's running time,* $O_\lambda(N \log N)$ *prover's running time, and a universal and updatable SRS.*

By applying the Fiat-Shamir transform, one can obtain a constant size SNARK, which is secure in the random oracle model assuming ARSDH. We recall the good, and often necessary practice, to instantiate the so-called strong Fiat-Shamir transform: hashing the common input with the current portion of the transcript to obtain the following challenge. As noted in [BPW12,HLPT20,DMWG23] the usage of weak Fiat-Shamir transform, where only the transcript is hashed, can be exploited to break adaptive soundness of the resulting SNARK.

References

ACK21. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 2021. Springer, Heidelberg. `doi:10.1007/978-3-030-84245-1_19`.

BB08. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008. `doi:10.1007/s00145-007-9005-7`.

BBB$^+$18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. `doi:10.1109/SP.2018.00020`.

BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018. doi:10.4230/LIPIcs.ICALP.2018.14.

BCG$^+$14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014. doi:10.1109/SP.2014.36.

BCGT13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC 2013*, pages 585–594, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. doi:10.1145/2488608.2488681.

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014. doi:10.1145/2591796.2591859.

BDFG20. Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020. https://eprint.iacr.org/2020/081.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_24.

BMM$^+$21. Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 65–97. Springer, Heidelberg, December 2021. doi:10.1007/978-3-030-92078-4_3.

BPW12. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 626–643. Springer, Heidelberg, December 2012. doi:10.1007/978-3-642-34961-4_38.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994. doi:10.1007/3-540-48658-5_19.

CFF$^+$21. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2021. doi:10.1007/978-3-030-92078-4_1.

CGKS23. Matteo Campanelli, Chaya Ganesh, Hamidreza Khoshakhlagh, and Janno Siim. Impossibilities in succinct arguments: Black-box extraction and more. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *AFRICACRYPT 23*, volume 14064 of *LNCS*, pages 465–489. Springer Nature, July 2023. doi:10.1007/978-3-031-37679-5_20.

CHM$^+$20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020. `doi:10.1007/978-3-030-45721-1_26`.

DMWG23. Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. Weak fiat-shamir attacks on modern proof systems. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 199–216. IEEE, 2023. `doi:10.1109/SP46215.2023.10179408`.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. `doi:10.1007/978-3-319-96881-0_2`.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. `doi:10.1007/3-540-47721-7_12`.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. `doi:10.1007/978-3-642-38348-9_37`.

GKM$^+$18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018. `doi:10.1007/978-3-319-96878-0_24`.

GR19. Alonso González and Carla Ràfols. Shorter pairing-based arguments under standard assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 728–757. Springer, Heidelberg, December 2019. `doi:10.1007/978-3-030-34618-8_25`.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. `doi:10.1007/978-3-662-49896-5_11`.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. `doi:10.1145/1993636.1993651`.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`.

HLPT20. Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *2020 IEEE Symposium on Security and Privacy*, pages 644–660. IEEE Computer Society Press, May 2020. `doi:10.1109/SP40000.2020.00048`.

JT20. Joseph Jaeger and Stefano Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In Rafael Pass and

Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 414–443. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64381-2_15.

KMS$^+$16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016. doi:10.1109/SP.2016.55.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11.

Lee21. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Heidelberg, November 2021. doi:10.1007/978-3-030-90453-1_1.

Lin01. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_10.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_10.

LM19. Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26948-7_19.

LPS23. Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic Group Model with Oblivious Sampling. In Guy Rothblum and Hoeteck Wee, editors, *TCC 2023 (4)*, volume 14372 of *LNCS*, pages 363–392, Taipei, Taiwan, Nov 29–Dec 2 2023. Springer, Cham. doi:10.1007/978-3-031-48624-1_14.

LSZ22. Helger Lipmaa, Janno Siim, and Michal Zajac. Counting vampires: From univariate sumcheck to updatable ZK-SNARK. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 249–278. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22966-4_9.

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_6.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. doi:10.1109/SP.2013.47.

PST13. Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_13.

RZ21. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84242-0_27.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

Sta21. StarkWare. ethSTARK documentation. Cryptology ePrint Archive, Report 2021/582, 2021. https://eprint.iacr.org/2021/582.

SZ20. Alan Szepieniec and Yuncong Zhang. Polynomial IOPs for Linear Algebra Relations. Technical Report 2020/1022, IACR, August 24, 2020. Last checked modification from June 9, 2021. URL: https://ia.cr/2020/1022.

TAB$^+$20. Alin Tomescu, Ittai Abraham, Vitalik Buterin, Justin Drake, Dankrad Feist, and Dmitry Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 45–64. Springer, Heidelberg, September 2020. doi:10.1007/978-3-030-57990-6_3.

Zha22. Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15982-4_3.

ZZ23. Cong Zhang and Mark Zhandry. The Relationship Between Idealized Models Under Computationally Bounded Adversaries Abstract. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023 (6)*, volume 14443 of *LNCS*, pages 390–419, Guangzhou, China, December 4–8, 2023. Springer, Cham. doi:10.1007/978-981-99-8736-8_13.

ZZK22. Cong Zhang, Hong-Sheng Zhou, and Jonathan Katz. An analysis of the algebraic group model. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 310–322. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22972-5_11.

A Expected versus Strict PPT Adversaries

In this section, we prove that if a falsifiable assumption is secure for strict PPT adversaries, it is also secure for expected PPT adversaries. Recall the definition of a falsifiable cryptographic assumption [GW11].

Definition 8. *A falsifiable cryptographic assumption consists of a PPT interactive challenger C and a constant $\epsilon \in [0, 1)$. On a security parameter λ, $C(1^\lambda)$ interacts with an adversary $\mathcal{A}(1^\lambda)$ and C outputs either* 0 *or* 1. *We say that $\mathcal{A}$ wins if the output is* 1.

The assumption related to (C, ϵ) states that for any efficient (either PPT or expected PPT) $\mathcal{A}$, we have $\Pr\left[\mathcal{A}(1^\lambda) \text{ wins } C(1^\lambda)\right] - \epsilon \approx_\lambda 0$, where the probability is over random coins of $\mathcal{A}$ and C.

For example, in decisional assumptions $\epsilon = 1/2$ and in search assumptions $\epsilon = 0$.

Lemma 6. *Let (C, ϵ) be a falsifiable assumption. If for all PPT $\mathcal{A}$, $\Pr\left[\mathcal{A}(1^\lambda) \text{ wins } C(1^\lambda)\right] - \epsilon \approx_\lambda 0$, then for any expected PPT $\mathcal{A}'$, $\Pr\left[\mathcal{A}'(1^\lambda) \text{ wins } C(1^\lambda)\right] - \epsilon \approx_\lambda 0$.*

Proof. Let $\varepsilon(\lambda)$ be a negligible function, such that $\Pr\left[\mathcal{A}(1^\lambda) \text{ wins } C(1^\lambda)\right] \leq \epsilon + \varepsilon(\lambda)$ for each PPT $\mathcal{A}$. Let T be the running time of $\mathcal{A}'$. Let $\mathcal{A}'$ be an expected PPT adversary and let T denote its running time. Suppose the expected running time $\mathbb{E}[T]$ is asymptotically bounded by $T_\mathbb{E}(\lambda)$ for some polynomial $T_\mathbb{E}(X)$.

By the Markov inequality, for each polynomial g, $\Pr[T \geq T_\mathbb{E}(\lambda) g(\lambda)] \leq \mathbb{E}[T]/(T_\mathbb{E}(\lambda) \cdot g(\lambda)) \leq 1/g(\lambda)$. Let $\mathcal{B}_{\mathcal{A}',g}$ be a (strict) PPT adversary that runs precisely as $\mathcal{A}$, except it halts and outputs $\perp$ if $\mathcal{A}$ has not terminated in $T_\mathbb{E}(\lambda) g(\lambda)$ steps. Conditioned on the event $T < T_\mathbb{E}(\lambda) g(\lambda)$, the output of $\mathcal{A}'$ and $\mathcal{B}_{\mathcal{A}',g}$ are equally distributed. We have

$$\begin{aligned}\Pr\left[\mathcal{A}'(1^\lambda) \text{ wins } C(1^\lambda)\right] \leq& \Pr\left[\mathcal{B}_{\mathcal{A}',g}(1^\lambda) \text{ wins } C(1^\lambda)\right] + \Pr\left[T \geq T_\mathbb{E}(\lambda) g(\lambda)\right] \\ \leq& \epsilon + \varepsilon(\lambda) + 1/g(\lambda) \ .\end{aligned}$$

Since the previous holds for any polynomial g, then $\Pr\left[\mathcal{A}'(1^\lambda) \text{ wins } C(1^\lambda)\right] - \epsilon - \varepsilon'(\lambda)$ is negligible. Moreover, since $\varepsilon'(\lambda)$ is negligible, and a sum of two negligible functions is negligible, then $\Pr\left[\mathcal{A}'(1^\lambda) \text{ wins } C(1^\lambda)\right] - \epsilon \approx_\lambda 0$. □

The previous lemma does not establish any tight relationship between the advantage of (strict) PPT adversaries and the advantage of expected PPT ones, besides the fact that if one of them is negligible, so is the other. The latter is not ideal since we reduce the advantage of (strict) PPT adversaries to concrete computational assumptions. To state concrete security parameters of our SNARK in Section 7, we want to reduce the advantage of expected PPT adversaries to the ARSDH assumption (that holds against strict PPT adversaries). We leave this as an important open question. We hope the techniques from [JT20] can be generalized to answer the previous question.

B AGMOS proof of ARSDH

While $\mathcal{S}$-RSDH is a known, falsifiable, standard-looking assumption, adaptive ARSDH is possibly stronger. Morover, [GR19] only gave an informal GGM (*without* oblivious sampling) argument why $\mathcal{S}$-RSDH is secure. To gain further confidence, we prove that ARSDH holds in the AGMOS [LPS23] under the PDL [Lip12] and Tensor Oracle FindRep (TOFR) [LPS23] assumptions. From this, it follows that ARSDH holds in the AGM under the PDL assumption. We emphasize that the security of our constructions (the KZG commitment and the resulting SNARKs) depend only on falsifiable ARSDH and not on the AGM(OS). We use AGMOS only as a sanity check.

Lipmaa et al. [LPS23] recently defined AGMOS (AGM with oblivious sampling). AGMOS is more realistic than AGM since AGMOS adversaries are given an additional power of sampling group elements without knowing their discrete logarithms. As shown in [LPS23], certain uses of KZG are secure in AGM but

```
$\mathcal{O}_\iota(E, D)$
────────────────────────────────
if $E \notin \mathcal{EF}_{\mathsf{p},\iota} \vee D \notin \mathcal{DF}_{\mathsf{p}}$ then return $\bot$; fi
$s \leftarrow_{\$} D$; $[\mathbb{q}]_\iota \leftarrow E(s)$; return $([\mathbb{q}]_\iota, s)$;
```

Fig. 7. The description of the oblivious sampling oracle $\mathcal{O}_\iota$, where $\iota \in \{1, 2\}$.

not in AGMOS. Since AGMOS is a new model, we will give a longer description of AGMOS and TOFR (an underlying security assumption); our description follows closely [LPS23].

Fix a pairing description $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$. Let $\mathcal{EF}_{\mathsf{p},\iota}$ be a set of (polynomially many) functions $\mathbb{F} \to \mathbb{G}_\iota$. Let $\mathcal{DF}_{\mathsf{p}}$ be a family of distributions over $\mathbb{F}$. We introduce two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$, one for each group $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$. The ith query (E, D) to $\mathcal{O}_\iota$ consists of a function $E \in \mathcal{EF}_{\mathsf{p},\iota}$ and a distribution $D \in \mathcal{DF}_{\mathsf{p}}$. The oracle samples a random field element $s_i \leftarrow_{\$} D$ and returns $[\mathbb{q}_{\iota_i}]_\iota \leftarrow E(s_i)$ and s_i.

We will denote the adversary's initial input (e.g., input from the challenger) in $\mathbb{G}_\iota$ by $[\mathbb{x}_\iota]_\iota$. We assume $[\mathbb{x}_\iota]_\iota$ always includes $[1]_\iota$. Let $\mathbb{x} = ([\mathbb{x}_1]_1, [\mathbb{x}_2]_2)$. The adversary's view consists of all group elements that the adversary has seen up to the given moment. This includes the adversary's initial input, elements sent by other parties during the interaction, and oracle answers.

Let $\mathcal{O}$ be as above. We require that for any PPT oracle adversary $\mathcal{A}^{\mathcal{O}}$, there exists a (non-uniform) PPT extractor $\mathsf{Ext}^{\mathcal{O}}_{\mathcal{A}}$, such that: if $\mathcal{A}^{\mathcal{O}}(\mathbb{x})$ outputs a vector of group elements $[\mathbb{y}]_\iota$, on input $\mathbb{x} = ([\mathbb{x}_1]_1, [\mathbb{x}_2]_2)$, then with an overwhelming probability, $\mathsf{Ext}^{\mathcal{O}}_{\mathcal{A}}$ outputs field-element matrices $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$, and $[\mathbb{q}_\iota]_\iota$ ($\mathcal{O}_\iota$'s answer vector), such that

$$\mathbb{y} = \boldsymbol{\gamma}^\intercal \mathbb{x}_\iota + \boldsymbol{\delta}^\intercal \mathbb{q}_\iota \ . \tag{5}$$

Definition 9 (AGMOS). *Let $\mathcal{EF} = \{\mathcal{EF}_{\mathsf{p},\iota}\}$ be a collection of functions. Let $\mathcal{DF} = \{\mathcal{DF}_{\mathsf{p}}\}$ be a family of distributions. A PPT algorithm $\mathcal{A}$ is an $(\mathcal{EF}, \mathcal{DF})$-*AGMOS adversary *for* Pgen *if there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$*, such that for any $\mathbb{x} = (\mathbb{x}_1, \mathbb{x}_2)$, $\mathsf{Adv}^{\mathrm{agmos}}_{\mathsf{Pgen},\mathcal{EF},\mathcal{DF},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l} \mathbb{y}_1 \neq \boldsymbol{\gamma}_1^\intercal \mathbb{x}_1 + \boldsymbol{\delta}_1^\intercal \mathbb{q}_1 \vee \\ \mathbb{y}_2 \neq \boldsymbol{\gamma}_2^\intercal \mathbb{x}_2 + \boldsymbol{\delta}_2^\intercal \mathbb{q}_2 \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); r \leftarrow \mathrm{RND}_\lambda(\mathcal{A}); \\ ([\mathbb{y}_1]_1, [\mathbb{y}_2]_2) \leftarrow_{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{p}, \mathbb{x}; r); \\ (\boldsymbol{\gamma}_\iota, \boldsymbol{\delta}_\iota, [\mathbb{q}_\iota]_\iota)_{\iota=1}^2 \leftarrow \mathsf{Ext}^{\mathcal{O}}_{\mathcal{A}}(\mathsf{p}, \mathbb{x}; r) : \end{array}\right] \approx_\lambda 0 \ .$$

$\mathcal{O}$ is the non-programmable oracle depicted in Fig. 7. Here, $[\mathbb{q}_\iota]_\iota$ is required to be the tuple of elements output by $\mathcal{O}_\iota$. We denote by il_ι the number of $\mathcal{O}_\iota$ calls.

Many AGMOS proofs rely on the following assumption.

Definition 10 (TOFR, **[LPS23]**). *Let $\mathcal{EF}$ be some family of function and $\mathcal{DF}$ a family of distributions. We say that* Pgen *is $(\mathcal{EF}, \mathcal{DF})$-*TOFR *(*Tensor Oracle FindRep*) secure if for any PPT $\mathcal{A}$,* $\mathsf{Adv}^{\mathrm{tofr}}_{\mathsf{Pgen},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\boldsymbol{v} \neq \mathbf{0} \wedge \boldsymbol{v}^\intercal \cdot \begin{pmatrix} 1 \\ \mathbb{q}_1 \\ \mathbb{q}_2 \\ \mathbb{q}_1 \otimes \mathbb{q}_2 \end{pmatrix} = 0 \middle| \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \boldsymbol{v} \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{p}) \right] \approx_\lambda 0 \ .$$

Here, $\mathcal{O}$, $\mathfrak{q}_1$, and $\mathfrak{q}_2$ are as in Definition 9.

Theorem 5. *Let $n \in \mathsf{poly}(\lambda)$ and* Pgen *be a bilinear-group generator. If $(n, 2)$-PDL and $(\mathcal{EF}, \mathcal{DF})$-*TOFR *hold, $(n+1)$-*ARSDH *holds in the $(\mathcal{EF}, \mathcal{DF})$-AGMOS.*

Proof (Sketch). Let $\mathcal{A}$ be an ARSDH adversary that with some non-negligible probability outputs $(\mathcal{S} = \{\alpha_i\}_{i=1}^{n+1}, [g, \varphi]_1)$ such that $|\mathcal{S}| = n+1$, $g \neq 0$, and $[g]_1 \bullet [1]_2 = [\varphi]_1 \bullet [\mathbf{Z}_{\mathcal{S}}(\sigma)]_2$. By using the algebraic representation provided the algebraic adversary, we obtain polynomials $g(X, \boldsymbol{Q})$ and $\varphi(X, \boldsymbol{Q})$, such that $[g]_1 = [g(\sigma, \mathfrak{q})]_1$ and $[\varphi]_1 = [\varphi(\sigma, \mathfrak{q})]_1$. Moreover, $g(X, \boldsymbol{Q}) = g_1(X) + \boldsymbol{g}_2^\intercal \boldsymbol{Q}$ and $\varphi(X, \boldsymbol{Q}) = \varphi_1(X) + \boldsymbol{\varphi}_2^\intercal \boldsymbol{Q}$ for some known coefficients. Define the verification polynomial

$$V(X, \boldsymbol{Q}) := g(X, \boldsymbol{Q}) - \varphi(X, \boldsymbol{Q})\mathbf{Z}_{\mathcal{S}}(X) = g_1(X) + \boldsymbol{g}_2^\intercal \boldsymbol{Q} - (\varphi_1(X) + \boldsymbol{\varphi}_2^\intercal \boldsymbol{Q})\mathbf{Z}_{\mathcal{S}}(X) \ .$$

Clearly, $V(X, \boldsymbol{Q}) = V^h(X) + V^t(X, \boldsymbol{Q})$ for $V^h(X) = g_1(X) - \varphi_1(X)\mathbf{Z}_{\mathcal{S}}(X)$ and $V^t(X, \boldsymbol{Q}) = \boldsymbol{g}_2^\intercal \boldsymbol{Q} - \boldsymbol{\varphi}_2^\intercal \boldsymbol{Q}\mathbf{Z}_{\mathcal{S}}(X)$. Here, V^h does not depend on $\boldsymbol{Q}$ while each term of V^t depends on some Q_k.

Importantly, we know all the coefficients of $V(X, \boldsymbol{Q})$. Since the verifier accepts, $V(\sigma, \mathfrak{q}) = 0$. Following [LPS23], we next consider separately four cases. The probability that $\mathcal{A}$ succeeds is bound by the sum of the probabilities it succeeds in all four cases.

<u>Case A ($V(X, \boldsymbol{Q}) = 0$ as a polynomial).</u> Then, $g_{2k} Q_k = \varphi_{2k} Q_k \mathbf{Z}_{\mathcal{S}}(X)$ for each k. Since each Q_k is an independent variable, we get that $g_{2k} = \varphi_{2k} = 0$. Thus, $g(X, \boldsymbol{Q}) = g_1(X)$ and $\varphi(X, \boldsymbol{Q}) = \varphi_1(X)$. Then, $\mathbf{Z}_{\mathcal{S}}(X) \mid g_1(X)$ and thus each α_i is a root of $g_1(X)$. By assumption, $\alpha_i \neq \alpha_j$. Hence, $g_1(X)$ has at least $n+1$ roots. Since $\deg g_1 \leq n$, $g_1(X)$ is a zero polynomial. Contradiction with $g_1(\sigma) = g \neq 0$.

<u>Case X.1 ($V(X, \boldsymbol{Q}) \neq 0$ but $V^t(X, \boldsymbol{Q}) = 0$).</u> Then, $V^h(X)$ is a non-zero polynomial with a known root σ. In this case, we construct a PDL adversary $\mathcal{B}$ that on input $\mathsf{ck} = ([(\sigma^i)_{i=0}^n]_1, [1, \sigma]_2)$ does the following. It sends ck to $\mathcal{A}$ and then obtains $(\mathcal{S}, [g, \varphi]_1)$. It uses the AGM extractor to obtain all coefficients of $g(X)$ and $\varphi(X)$. It factors $V^h(X)$, finding its at most $2n+1$ roots x_i. It then tests by brute force for each i whether $[\sigma]_1 = [x_i]_1$, and returns the value $\sigma \leftarrow x_j$ for which $[\sigma]_1 = [x_j]_1$ holds. Clearly, $\mathcal{B}$ succeeds in computing σ with the same probability as $\mathcal{A}$ succeeds in breaking the ARSDH.

<u>Case X.2 ($V^t(X, \boldsymbol{Q}) \neq 0$ but $V^t(\sigma, \boldsymbol{Q}) = 0$).</u> Recall

$$V^t(X, \boldsymbol{Q}) = \sum (g_{2k} - \varphi_{2k}\mathbf{Z}_{\mathcal{S}}(X))Q_k = \sum \beta_k(X) Q_k \ ,$$

where $\beta_k(X) := g_{2k} - \varphi_{2k}\mathbf{Z}_{\mathcal{S}}(X)$. In Case X.2, we have that $\beta_k(X) \neq 0$ but $\beta_k(\sigma) = 0$ for some k. Hence, σ is the root of the non-zero polynomial $\varphi_{2k}\mathbf{Z}_{\mathcal{S}}(X) - g_{2k}$. One can construct a different PDL adversary that computes σ.

<u>Case Q ($V^t(\sigma, \boldsymbol{Q}) \neq 0$ but $V^t(\sigma, \mathfrak{q}) \neq 0$).</u> We can construct a TOFR adversary that samples σ, constructs ck, and after interacting with $\mathcal{A}$, returns

$$\boldsymbol{v} = \begin{pmatrix} V^h(\sigma) \\ \boldsymbol{\beta}(\sigma) \\ \mathbf{0} \end{pmatrix} \ .$$

Since $\boldsymbol{v}$ is a non-zero vector and

$$\boldsymbol{v}^\top \cdot \begin{pmatrix} 1 \\ \mathfrak{q}_1 \\ \mathfrak{q}_2 \\ \mathfrak{q}_1 \otimes \mathfrak{q}_2 \end{pmatrix} = 0 \enspace ,$$

$\mathcal{B}$ breaks the TOFR assumption.

Summarizing, if TOFR and PDL hold, ARSDH is secure in the AGMOS. □

C The Relation of BBE and Evaluation Binding

It is always important to understand how novel security notions related to each other and already known security notions. We prove that if black-box extractability holds, then (a variant of) evaluation binding follows from (and, thus, is equivalent to) binding. We only obtain a variant (see Definition 11) due to the same issue as with the definition of black-box extractability itself: namely, α has to be chosen randomly and not by the adversary.

Definition 11. *A polynomial commitment scheme* PC *is* non-adaptively evaluation-binding *for* Pgen*, if for any* λ*,* $n \in \mathsf{poly}(\lambda)$*, and PPT adversary* $\mathcal{A}$*, DPT* P^**, and* $\mathcal{F}_\mathsf{p} \subseteq \mathbb{F}$ *of size* $\lambda^{\omega(1)}$*,* $\mathsf{Adv}^{\mathrm{naeb}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A},\mathsf{P}^*}(\lambda) :=$

$$\Pr\left[\begin{array}{l|l} \mathsf{V}(\mathsf{ck}, C, \alpha, \eta, \pi) = 1 \wedge & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ \mathsf{V}(\mathsf{ck}, C, \alpha, \eta', \pi') = 1 \wedge & (C, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{p}, \mathsf{ck}); \alpha \leftarrow_\$ \mathcal{F}_\mathsf{p}; \\ \eta \neq \eta' & (\eta, \pi, \eta', \pi') \leftarrow \mathsf{P}^*(\mathsf{st}, \alpha) \end{array}\right] \approx_\lambda 0 \enspace .$$

We also recall the standard binding definition for (non-hiding) polynomial commitment schemes.

Definition 12. *A polynomial commitment scheme* PC *is* binding *for* Pgen*, if for any* λ*,* $n \in \mathsf{poly}(\lambda)$ *and PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{bind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l|l} C = \mathsf{Com}(f_1) = \mathsf{Com}(f_2) \wedge & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ f_1 \neq f_2 & (C, f_1, f_2) \leftarrow \mathcal{A}(\mathsf{p}, \mathsf{ck}) \end{array}\right] \approx_\lambda 0 \enspace .$$

Theorem 6. *If a non-interactive PCS* PC *is black-box extractable and binding, it is non-adaptively evaluation-binding.*

Proof. Let $(\mathcal{A}_{eb}, \mathsf{P}^*_{eb})$ be some arbitrary adversaries against non-adaptive evaluation-binding. We construct a binding adversary $\mathcal{A}_{\mathsf{bind}}$ and black-box extractability adversaries $(\mathcal{B}_i, \mathsf{P}^*_i)$, $i \in \{1, 2\}$, used by $\mathcal{A}_{\mathsf{bind}}$. Since PC is black-box extractable, there exists an expected PPT extractor $\mathsf{Ext}_{\mathsf{bb}}$ that works for $(\mathcal{B}_i, \mathsf{P}^*_i)$, $i \in \{1, 2\}$.

We first describe $(\mathcal{B}_i, \mathsf{P}^*_i)$, $i \in \{1, 2\}$, that are given (C, α) as an input. First, $\mathcal{B}_i(\mathsf{p}, \mathsf{ck})$ is just the same as $\mathcal{A}_{eb}$, and returns (C, st). Second, $\mathsf{P}^*_i(\mathsf{st}, \alpha)$ runs $\mathsf{P}^*_{eb}(\mathsf{st}, \alpha)$, obtaining either $(\eta_1 \neq \eta_2, \pi_1, \pi_2)$ or $\perp$. If P^*_{eb} succeeds in producing a collision, P^*_i returns $\mathsf{tr}_i \leftarrow (C, \alpha, \eta_i, \pi_i)$. Otherwise, P^*_i returns $\perp$.

Since PC is black-box extractable, $\mathsf{Ext}_{\mathsf{bb}}^{\mathsf{P}^*_i(\mathsf{st},\cdot)}(\mathsf{ck}, \mathsf{tr}_i)$ outputs a polynomial f_i, such that if V accepts tr_i then, with an overwhelming probability, $C = \mathsf{Com}(f_i)$ and $f_i(\alpha) = \eta_i$.

Finally, $\mathcal{A}_{\mathsf{bind}}(\mathsf{ck})$ does the following:

1. Invoke $\mathcal{A}_{eb}(\mathsf{ck}) = \mathcal{B}_1(\mathsf{p}, \mathsf{ck}) = \mathcal{B}_2(\mathsf{p}, \mathsf{ck})$, obtaining (C, st).
2. Sample $\alpha \leftarrow_{\$} \mathbb{F}$.
3. For $i \in \{1, 2\}$, run $\mathcal{B}_i(\mathsf{ck}, C, \alpha)$ and $f_i \leftarrow \mathsf{Ext}_{\mathsf{bb}}^{\mathsf{P}_i^*(\mathsf{st},\cdot)}(\mathsf{ck}, \mathsf{tr}_i)$ with P_i^*. Return $\perp$ if $f_1 = \perp$, $f_2 = \perp$, or $f_1 = f_2$.
4. Return (C, f_1, f_2).

$\mathcal{A}_{\mathsf{bind}}$ fails to break binding when,

– Extraction of either of the polynomials f_i fails, or
– $\mathcal{A}_{eb}$ fails to break evaluation binding.

Therefore,

$$1 - \mathsf{Adv}^{\text{bind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}_{\mathsf{bind}}}(\lambda) \leq \mathsf{Adv}^{\text{bbe}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext},n,\mathcal{B}_1,\mathsf{P}_1^*}(\lambda) + \mathsf{Adv}^{\text{bbe}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext},n,\mathcal{B}_2,\mathsf{P}_2^*}(\lambda) + \\ (1 - \mathsf{Adv}^{\text{naeb}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A},\mathsf{P}^*}(\lambda)) \enspace .$$

Consequently,

$$\mathsf{Adv}^{\text{bind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}_{\mathsf{bind}}}(\lambda) + \mathsf{Adv}^{\text{bbe}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext},n,\mathcal{B}_1,\mathsf{P}_1^*}(\lambda) + \mathsf{Adv}^{\text{bbe}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext},n,\mathcal{B}_2,\mathsf{P}_2^*}(\lambda) \\ > \mathsf{Adv}^{\text{naeb}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A},\mathsf{P}^*}(\lambda) \enspace .$$

However, note that $\mathcal{A}_{\mathsf{bind}}$ is an expected PPT adversary, not a strict PPT adversary, as the binding definition requires. Here, we can apply Lemma 6. Since binging holds against all PPT adversaries, it also holds against all expected PPT adversaries.

Therefore, $\mathsf{Adv}^{\text{bind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}_{\mathsf{bind}}}(\lambda)$ is negligible and so are $\mathsf{Adv}^{\text{bbe}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext},n,\mathcal{B}_i,\mathsf{P}_i^*}(\lambda)$ for $i \in \{1, 2\}$ because BBE holds. It follows that $\mathsf{Adv}^{\text{naeb}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A},\mathsf{P}^*}(\lambda)$ is negligible. $\square$